



Developing Burp Suite Extensions

From manual testing to security automation

Luca Caretoni - luca@doyensec.com

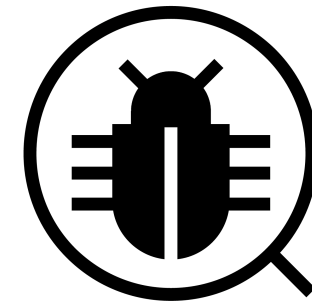
Welcome

- This is a brand-new class!
- Hands-on and highly interactive
- We have 14 hours to go from manual testing to security automation

Ingredients



Jenkins



Agenda - Day 1

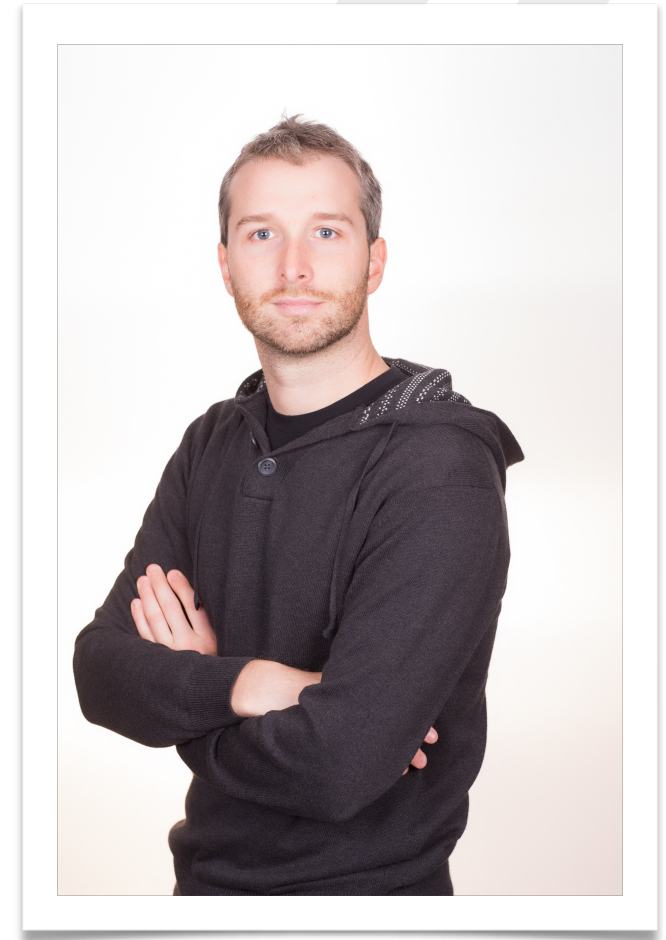
1. Intro
2. A quick recap of Burp Suite's tools
3. Understanding Burp's Extensibility APIs
4. IDE setup and templates
5. *Hello Burp* extension
6. Customer logger

Agenda - Day 2

7. Replay tool
8. Passive check for the scanner
 - Detect pages missing SRI attribute
9. Active check for the scanner
 - Detect EJL vulnerabilities
10. Security automation toolchain integrated in Jenkins
11. Intruder payload generator case study (Bradamsa)

About me

- ♥ AppSec since 2004
- Doyensec Co-founder
- Former AppSec @LinkedIn, Director of Security (Addepar), Senior Security Researcher (Matasano),



Relevant work

- First edition of **Instant Burp Suite Starter**
 - Packt Publishing, 70 pages, ISBN-10 1849695180
- Numerous extensions:
 - Bradamsa - <https://github.com/ikkisoft/bradamsa>
 - Blazer <https://github.com/ikkisoft/blazer>
 - ParrotNG <https://github.com/ikkisoft/ParrotNG>
 - etc..
- Burp Suite's anti-debugging patch me
 - <http://blog.nibblesec.org/2013/01/anti-debugging-techniques-and-burp-suite.html>

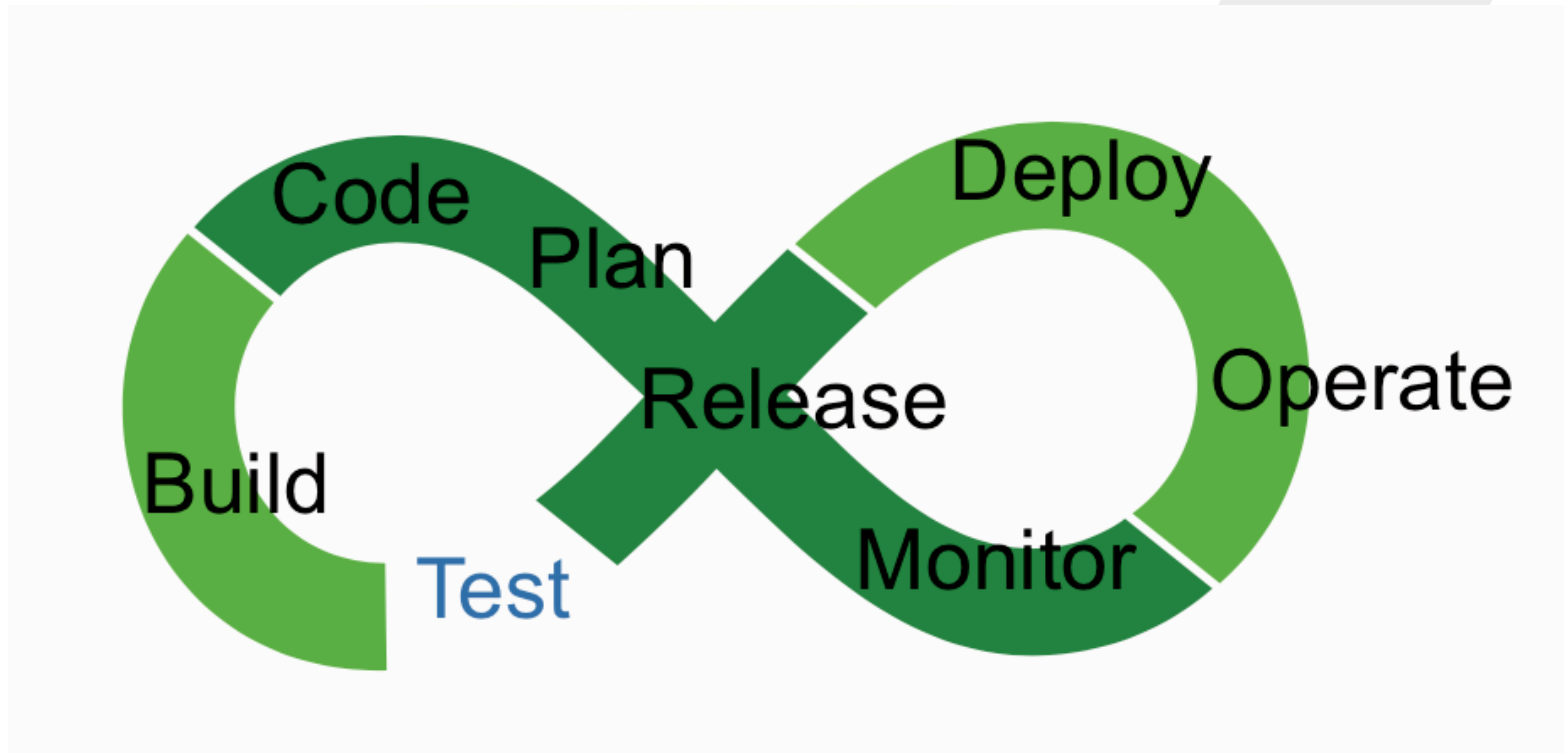
Contact Details

- If you have questions, problems - even after the training:
 - Email: luca@doyensec.com
 - Twitter: [@lucacarettoni](https://twitter.com/lucacarettoni)

Getting to know you

- Builder or Breaker
 - How many pentesters?
 - How many software engineers?
 - ...
- Have you ever used Burp? How often?
- Let's talk about coding in Java
 - *Hello World* or *Cocktail Shaker Sort* algorithm?

Tactical Testing



The background is a dark grey gradient. At the top, there is a pattern of binary code (0s and 1s) in a light grey color. On the right side, there is a large, stylized, multi-layered 'D' shape in shades of grey. On the left side, there is a large orange triangle pointing towards the center.

Setup

Getting all ready...

Java / Python / Ruby

- I will be coding in Java with Netbeans
- You can follow examples in any of the three languages
- For hands-on exercises, I would highly recommend coding in Java too!

What you need

- Laptop :)
- Recent Java JDK installed
- Oracle Netbeans IDE installed
 - You can use Eclipse or IDEA (if you're very familiar with those)
- Git command line installed

Burp Suite Professional

- Since we're building Scanner extensions, **this training requires a Pro license**
- You can either use your own license, or I can provide a temporary license kindly offered by



Burp Setup

- Configure your browser to use the Burp Proxy listener as its HTTP proxy server
- *Firefox* is probably the best browser for security testing

Setup Proxy in Internet Explorer

- Go to the Tools menu, select Internet Options, go to the Connections tab, and click on the "LAN settings" button. Make sure the "Automatically detect settings" box is unchecked. Make sure the "Use automatic configuration script" box is unchecked. Make sure the "Use a proxy server for your LAN" box is checked. Enter your Burp Proxy listener address in the "Address" field (by default, 127.0.0.1). Enter your Burp Proxy listener port in the "Port" field (by default, 8080). Make sure the "Bypass proxy server for local addresses" box is unchecked. Then click on the "Advanced" button. Make sure the "Use the same proxy server for all protocols" box is checked. Delete anything that appears in the "Exceptions" field. Then click "OK" to close all of the options dialogs.

Setup Proxy in Chrome

- The Chrome browser picks up the HTTP proxy settings configured on the host computer. If you are using Chrome, you can open your computer's built-in browser and follow the instructions for configuring that. If you aren't sure where the built-in proxy settings are, open Chrome, go to the Customize menu, select Settings, click on "Show advanced settings", and click the "Change proxy settings ..." button. This will open the relevant configuration options for your host computer.

Setup Proxy in Firefox

- Go to the Firefox menu, click on Options, click on Advanced, go to the Network tab, and click on the Settings button in the Connection section. Select the "Manual proxy configuration" radio button. Enter your Burp Proxy listener address in the "HTTP proxy" field (by default, 127.0.0.1). Enter your Burp Proxy listener port in the "Port" field (by default, 8080). Make sure the "Use this proxy server for all protocols" box is checked. Delete anything that appears in the "No proxy for" field. Then click "OK" to close all of the options dialogs.

Setup Proxy in Safari

- Go to the Safari menu, click on Preferences, click on Advanced, and by the Proxies label click the "Change Settings" button. This will open the Network configuration settings for your current network adapter. In the Proxies tab, check the "Web Proxy (HTTP)" box, and enter your Burp Proxy listener address in the "Web Proxy Server" field (by default, 127.0.0.1), and your Burp Proxy listener port in the (unlabeled) port field (by default, 8080). Ensure the "Bypass proxy settings for these Hosts & Domains" box is empty. Repeat these steps for the "Secure Web Proxy (HTTPS)" checkbox. Click "OK" and "Apply" and close the open dialogs.

Clone the training repo

1. Open the terminal
2. From your working directory, clone using

```
$ git clone https://github.com/  
doyensec/burpdeveltraining.git
```

Training Repo Structure

- ▶ BurpExtensionTemplate
 - ▶ Eclipse
 - ▶ IDEA
 - ▶ Netbeans
- ▶ Bradamsa
- ▶ DetectELJ
 - ▶ Java
 - ▶ Python
 - ▶ Ruby
- ▶ DetectSRI
 - ▶ Java
 - ▶ Python
 - ▶ Ruby
- ▶ HelloBurp
 - ▶ Java
 - ▶ Python
 - ▶ Ruby
- ▶ ReplayAndDiff
 - ▶ Java
 - ▶ Python
 - ▶ Ruby
- ▶ SiteLogger
 - ▶ Java
 - ▶ Python
 - ▶ Ruby

Each extension includes **Work In Progress** versions:

- /WIP1/
- /WIP2/
- /Final/

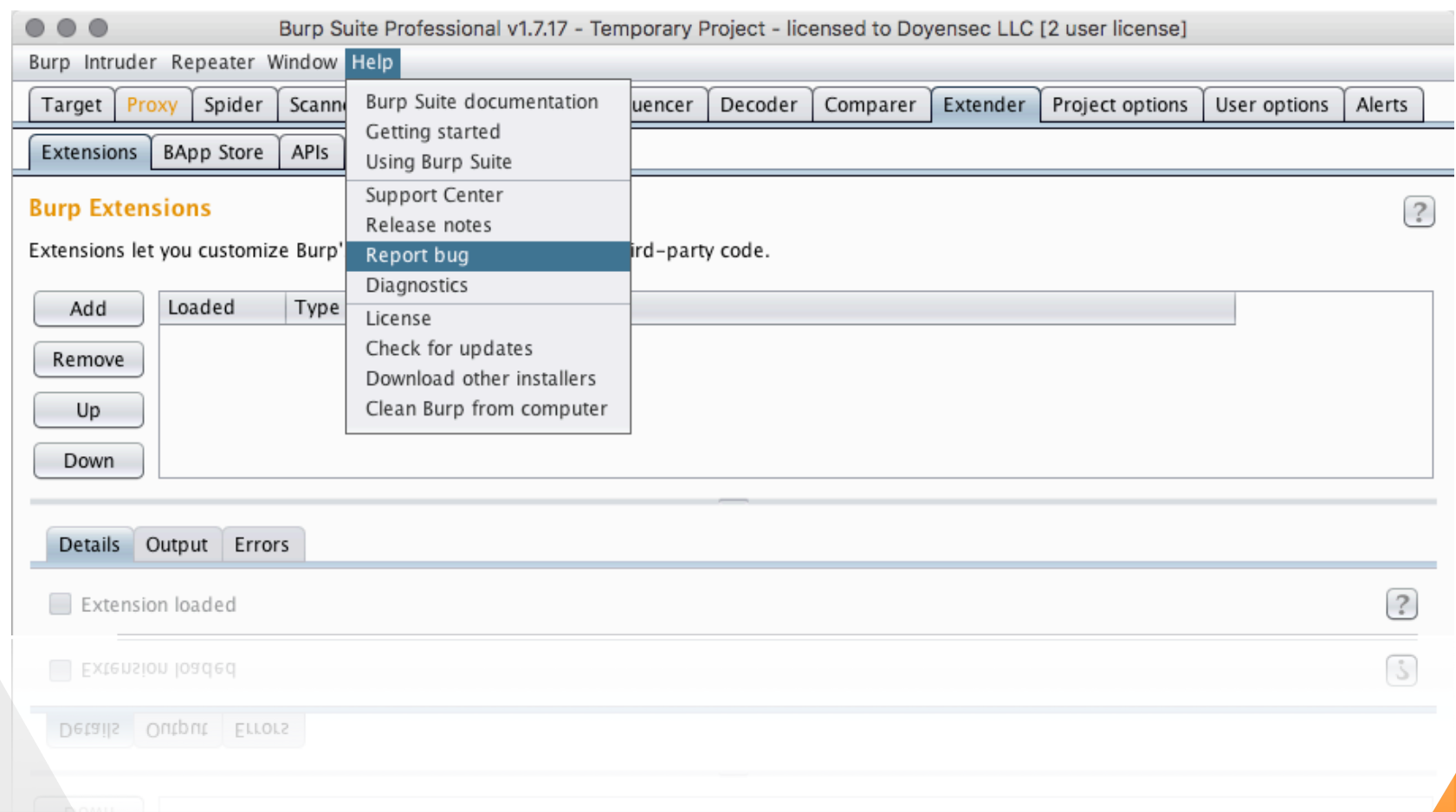
BURPSUITE

- Burp is the Swiss-army knife of Web Application Security
- Multiple tools, seamlessly integrated
 - Proxy
 - Spider
 - Scanner (Pro version only)
 - Intruder
 - Repeater
 - Sequencer
 - Decoder
 - Comparer

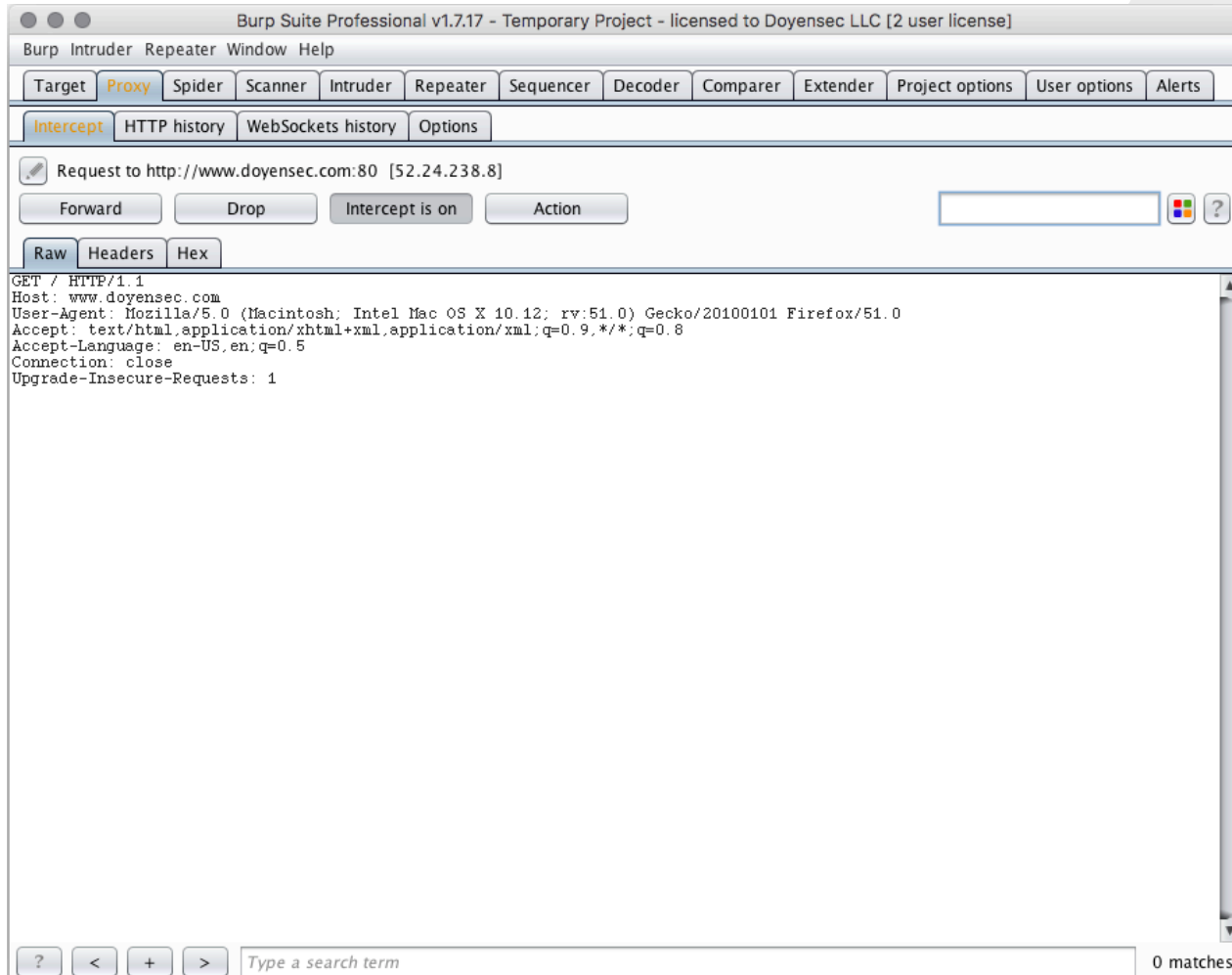
From Burp v1.0 - June 2003



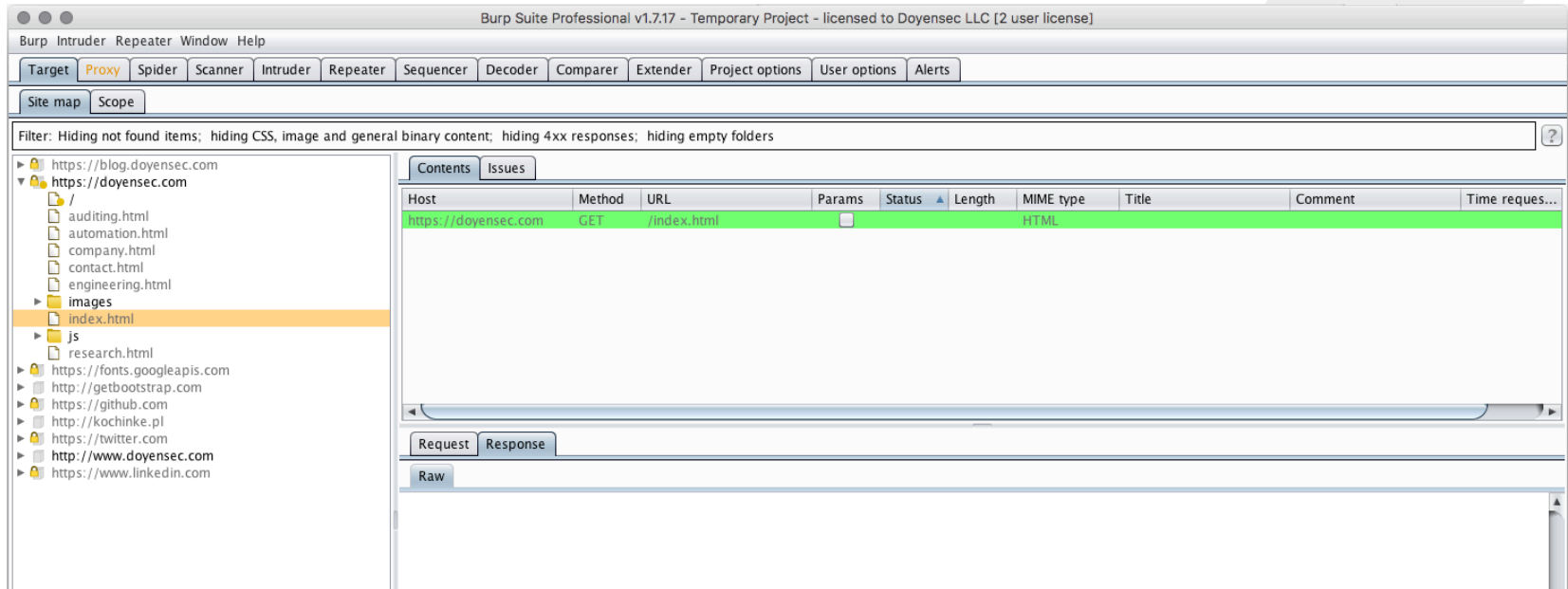
To v1.7.17 and counting...



Proxy



Target



Spider

The screenshot shows the Burp Suite Professional v1.7.17 interface. The main window is titled "Burp Suite Professional v1.7.17 - Temporary Project - licensed to Doyenssec LLC [2 user license]". The "Spider" tab is selected in the top menu. Below the menu, there are tabs for "Control" and "Options".

Spider Status

Use these settings to monitor and control Burp Spider. To begin spidering, browse to the target application, then right-click one or more nodes in the target site map, and choose "Spider this".

Buttons: **Spider is running** (active), **Clear queues**

Requests made: 196
Bytes transferred: 2,016,077
Requests queued: 702
Forms queued: 1

Spider Scope

- Use suite scope [defined in Target tab]
- Use custom scope

Burp Spider - Submit Form

Burp Spider needs your guidance to submit a login form. Please choose the value of each form field which should be used when submitting the form. You can control how Burp handles forms in the Spider options tab.

Action URL: `https://www.troopers.de/login/?next=/account/`
Method: POST

Type	Name	Value
Hidden	csrfmiddlewaretoken	dRkyPfANtcMaVAJ3EXMevosOrVEbVHa8
Password	password	
Text	username	

Buttons: **Submit form**, **Ignore form**

Scanner

Burp Suite Professional v1.7.17 - Temporary Project - licensed to Doyensec LLC [2 user license]

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts

Issue activity Scan queue Live scanning Issue definitions Options

#	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confid
6	18:25:24 22 Feb 2017	Issue found	i Email addresses disclosed	https://doyensec.com	/		Information	Certa
7	18:25:24 22 Feb 2017	Issue found	i HTML uses unrecognized charset	https://doyensec.com	/		Information	Tenta
8	18:31:27 22 Feb 2017	Issue found	w SSL certificate	https://www.troopers.de	/		Medium	Certa
9	18:31:27 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/troopers17/		Information	Certa
10	18:31:27 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/troopers-logo.svg		Information	Certa
11	18:31:27 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/ernw-logo.svg		Information	Certa
12	18:31:28 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/icon_attack-and-research.svg		Information	Certa
13	18:31:28 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/icon_defense-and-management.svg		Information	Certa
14	18:31:28 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/icon_sap-security.svg		Information	Certa
15	18:31:29 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/images/conference-overview-2017.svg		Information	Certa
16	18:31:29 22 Feb 2017	Issue found	i Content type is not specified	https://www.troopers.de	/static/fonts/source-sans-pro-v9-latin-regular.woff2		Information	Certa
17	18:31:29 22 Feb 2017	Issue found	i Content type is not specified	https://www.troopers.de	/static/fonts/exo-2-v3-latin-regular.woff2		Information	Certa
18	18:31:29 22 Feb 2017	Issue found	i Content type is not specified	https://www.troopers.de	/static/fonts/source-sans-pro-v9-latin-700.woff2		Information	Certa
19	18:31:29 22 Feb 2017	Issue found	i Cacheable HTTPS response	https://www.troopers.de	/static/fonts/fontawesome-webfont.woff		Information	Certa
20	18:31:29 22 Feb 2017	Issue found	w Content type incorrectly stated	https://www.troopers.de	/static/fonts/fontawesome-webfont.woff		Low	Firm

Advisory Request Response

i Cacheable HTTPS response

Issue: Cacheable HTTPS response
 Severity: Information
 Confidence: Certain
 Host: https://www.troopers.de
 Path: /static/images/troopers-logo.svg

Issue description

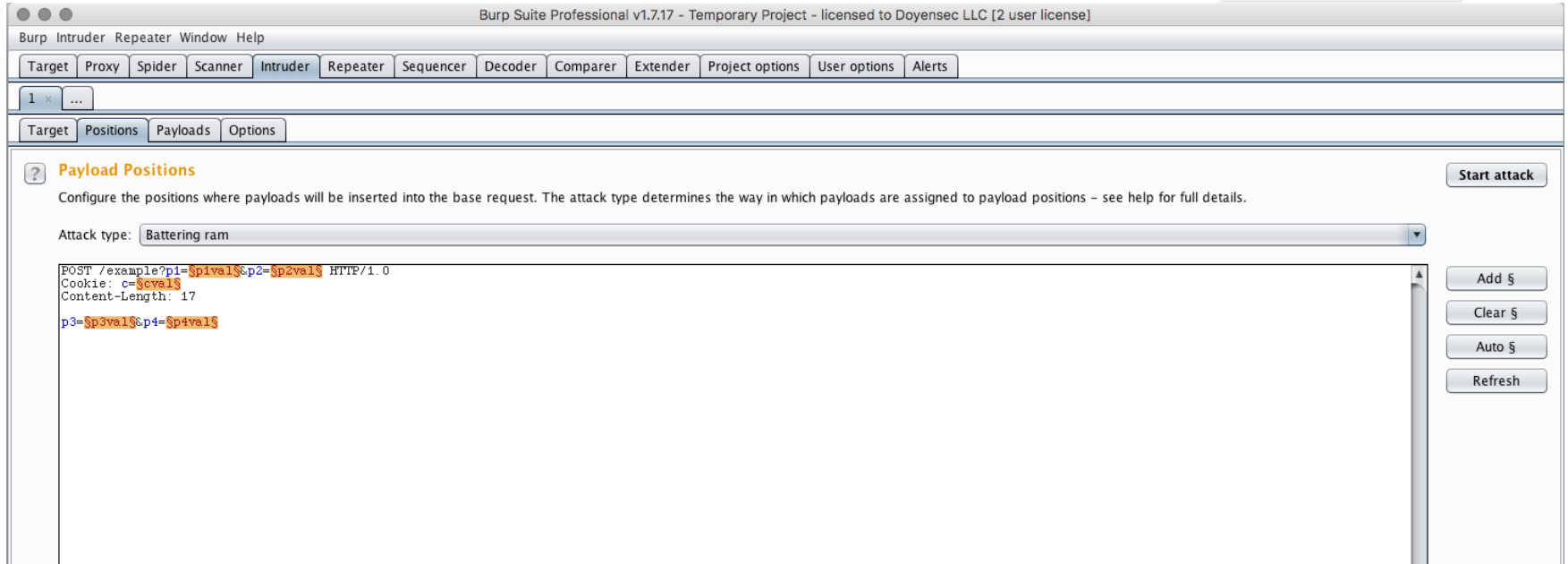
Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.

Issue remediation

Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root. Alternatively, most web development platforms allow you to control the server's caching directives from within individual scripts. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

- Cache-control: no-store
- Pragma: no-cache

Intruder



Repeater

Burp Suite Professional v1.7.17 - Temporary Project - licensed to Doyenssec LLC [2 user license]

Target: https://doyenssec.com

Request

```
GET /index.html HTTP/1.1
Host: doyenssec.com
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response

```
Value
... 200 OK
... Wed, 22 Feb 2017 17:38:33 GMT
... Apache/2.4.18 (Ubuntu)
... Tue, 21 Feb 2017 22:33:46 GMT
... "3861-54911f91c9ab6"
... bytes
... 14433
... Accept-Encoding
... close
... text/html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset="utf-8" />
  <!--[if IE]><meta http-equiv="X-UA-Compatible" content="IE=edge, chrome=1" /><![endif]-->
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="description" content="Doyenssec is an independent security research and development company focused on vulnerability discovery and remediation." />
  <meta name="keywords" content="doyenssec, doyenssecurity, security, infosec, vulnerability, exploit, offensive, pentest, pentester, hacking, auditing, reverse engineering, automation" />
  <meta name="robots" content="index, follow" />
  <meta name="robots" content="noodp" />
  <meta name="robots" content="noydir" />
  <title>Doyenssec :: Build with Security</title>
  <link href="css/bootstrap.min.css" rel="stylesheet" type="text/css" />
```

Sequencer

Burp Suite Professional v1.7.17 - Temporary Project - licensed to Doyensec LLC [2 user license]

Target Proxy Spider Scanner Intruder Repeater **Sequencer** Decoder Comparer Extender Project options User options Alerts

Live capture Manual load Analysis options

Select Live Capture Request

Send requests here from other tools to configure a live capture. Select the request to use, configure the other options below, then click "Start live capture".

#	Host	Request
1	https://doyenssec.com	GET /index.html HTTP/1.1 Host: doyenssec...

Start live capture

Token Location Within Response

Select the location in the response where the token appears.

Cookie:

Form field:

Custom location:

Live Capture Options

These settings control the engine used for making HTTP requests and handling responses.

Number of threads:

Throttle between requests (milliseconds):

Ignore tokens whose length deviates by characters

Burp Sequencer [live capture #1: https://doyenssec.com]

Live capture (paused)

Resume Copy tokens Auto analyze (next: 200) Requests: 139

Stop Save tokens Analyze now Errors: 0

Summary Character-level analysis Bit-level analysis Analysis Options

Overall result

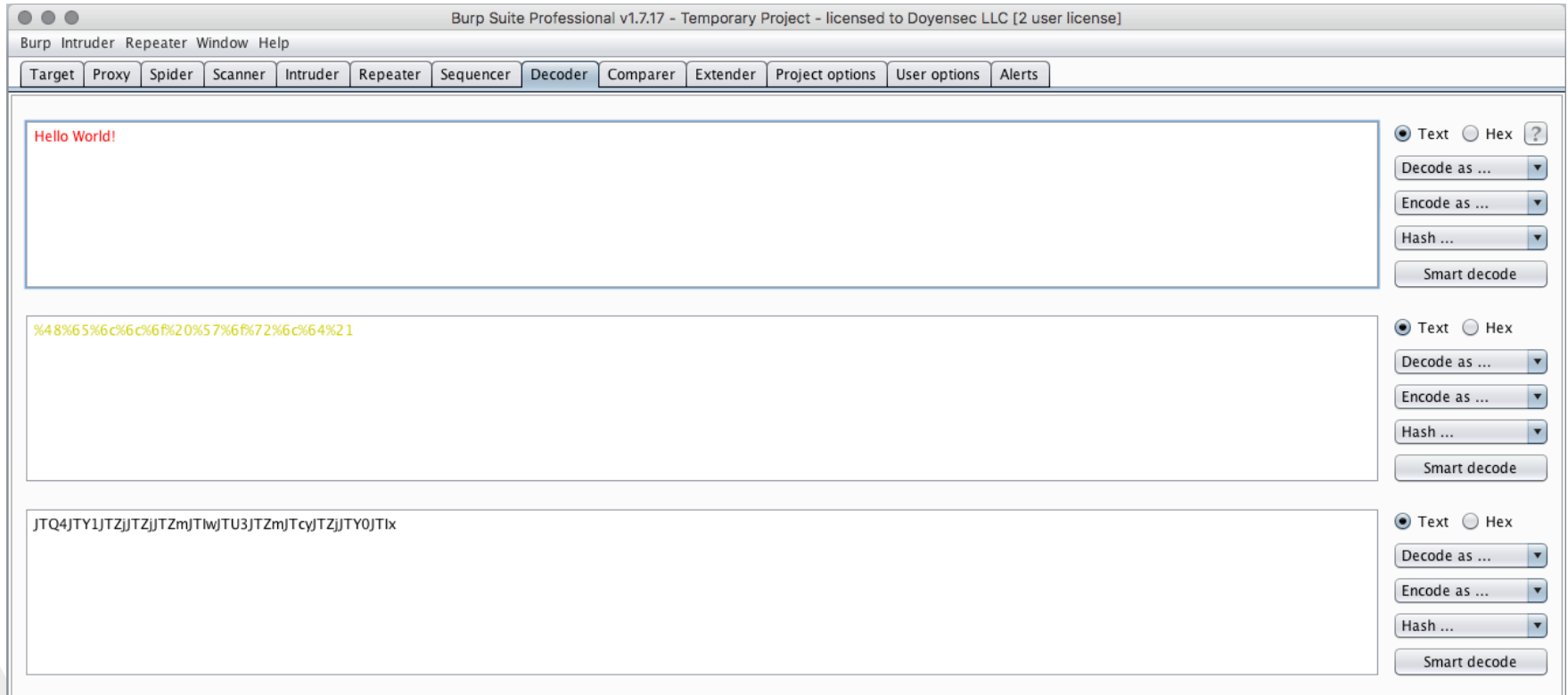
The overall quality of randomness within the sample is estimated to be: extremely poor.
At a significance level of 1%, the amount of effective entropy is estimated to be: 0 bits.

Note: Character-level analysis was not performed because the sample size is too small relative to the size of the character set used in the sampled tokens.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

Decoder



Comparer

The screenshot shows the Burp Suite Professional v1.7.17 interface. The 'Comparer' window is active, displaying a comparison of two items. The first item is selected and highlighted in yellow. The comparison shows 0 differences between the two items.

Comparer

This function lets you do a word- or byte-level comparison between different data. You can load, paste, or send data here from other tools and then select the comparison you want to perform.

Select item 1:

#	Length	Data
1	35	https://www.troopers.de/troopers17/
2	35	https://www.troopers.de/troopers17/

Buttons: Paste, Load, Remove, Clear

Byte compare of #1 and #2 (0 differences)

Length: 35 Text Hex Length: 35

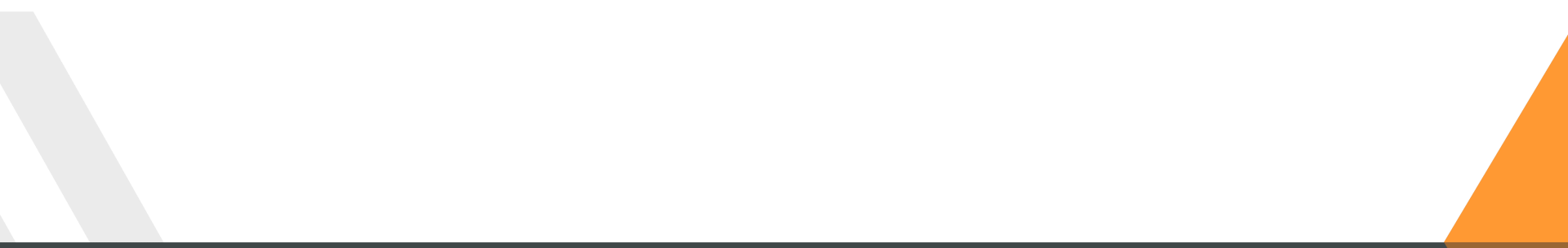
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
0	68	74	74	70	73	3a	2f	2f	77	77	77	2e	74	72	6f	6f	https://www.troopers.de/troopers																			
1	70	65	72	73	2e	64	65	2f	74	72	6f	6f	70	65	72	73	pers.de/troopers																			
2	31	37	2f	--	--	--	--	--	--	--	--	--	--	--	--	--	17/																			

Burp Suite Trivia 1/2

- One question, twenty seconds and win Burp Suite merchandise!
- No Googling!



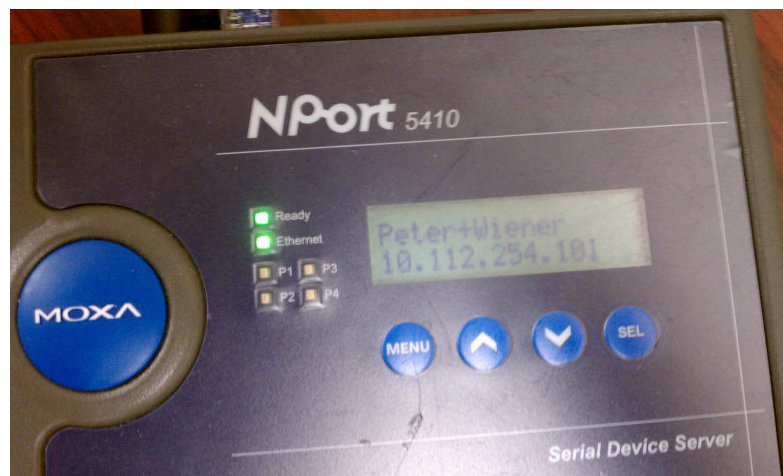
- What's the default name used by Burp for form submissions?



and the two answers are...

- Peter Wiener
- Peter Winter

- Also, don't spider network devices!



Extender

- Lets you extend the functionalities of Burp in many ways
- Available since v1.1 (although radically different). Starting from v1.5.01, as known today
- Extensions can be written in *Java*, *Python* (*Jython*) or *Ruby* (*jRuby*)

Free vs Pro

- Extensions are compatible for both Burp Suite Free and Professional versions

	Free Edition	Professional Edition \$349 per user per year
Burp Proxy	✓	✓
Burp Spider	✓	✓
Burp Repeater	✓	✓
Burp Sequencer	✓	✓
Burp Decoder	✓	✓
Burp Comparer	✓	✓
Burp Intruder	? Time-throttled demo	✓
Burp Scanner	?	✓
Save and Restore	?	✓
Search	?	✓
Target Analyzer	?	✓
Content Discovery	?	✓
Task Scheduler	?	✓
Release Schedule	? Major point releases	Frequent updates, earlier releases, beta versions

Core Principles

- Allow multiple extensions to run simultaneously
- Support dynamic loading and unloading of extensions at runtime
- Support languages other than Java
- Provide a much, much richer API that allows extensions to really integrate with Burp's internals
- Use a more future-proof API design, to allow easier enhancements in future
- As far as possible, ensure backwards compatibility with legacy extensions

From <http://blog.portswigger.net/2012/12/new-burp-suite-extensibility.html>

Extender GUI 1/4

- Load/Unload extensions
- StdIn and Stdout (Console, File, Show in UI)

Extensions BApp Store APIs Options

Burp Extensions

Extensions let you customize Burp's behavior using your own or third-party code.

Add Remove Up Down

Loaded	Type	Name
<input checked="" type="checkbox"/>	Java	Bradamsa

Details Output Errors

Extension loaded

Name: Bradamsa

Item	Detail
Extension type	Java
Filename	bapps/cfba9ea9b11f4436a4303bca58b8ff06/bradamsa.jar
Method	registerExtenderCallbacks
Suite tabs	1
Intruder payload generators	1

Extender GUI 2/4

- Burp App Store (BApp)
- Available since March 2014
- All BApp extensions source code is available on Github - see <https://portswigger.net/bappstore/>

The screenshot shows the Burp App Store interface. At the top, there are tabs for 'Extensions', 'BApp Store', 'APIs', and 'Options'. The 'BApp Store' tab is selected. Below the tabs, the title 'BApp Store' is displayed, followed by a description: 'The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.'

A table lists various extensions with columns for Name, Installed, Rating, and Detail. The 'Yara' extension is highlighted in the table. The 'Detail' column for 'Yara' shows a description, requirements, author information, and a rating.

Name	Installed	Rating	Detail
.NET Beautifier	<input type="checkbox"/>	★★★★★	
Active Scan++	<input type="checkbox"/>	★★★★★	
Additional Scanner Checks	<input type="checkbox"/>	★★★★★	
AES Payloads	<input type="checkbox"/>	★★★★★	Pro extension
AuthMatrix	<input type="checkbox"/>	★★★★★	
Authz	<input type="checkbox"/>	★★★★★	
Autorize	<input type="checkbox"/>	★★★★★	
Backslash Powered Scanner	<input type="checkbox"/>	★★★★★	Pro extension
Batch Scan Report Generator	<input type="checkbox"/>	★★★★★	Pro extension
Blazer	<input type="checkbox"/>	★★★★★	
Bradamsa	<input checked="" type="checkbox"/>	★★★★★	
Browser Repeater	<input type="checkbox"/>	★★★★★	
Buby	<input type="checkbox"/>	★★★★★	
Burp Chat	<input type="checkbox"/>	★★★★★	
Burp CSJ	<input type="checkbox"/>	★★★★★	
Burp-hash	<input type="checkbox"/>	★★★★★	Pro extension
BurpSmartBuster	<input type="checkbox"/>	★★★★★	
Bypass WAF	<input type="checkbox"/>	★★★★★	
Carbonator	<input type="checkbox"/>	★★★★★	
CO2	<input type="checkbox"/>	★★★★★	
Code Dx	<input type="checkbox"/>	★★★★★	
Commentator	<input type="checkbox"/>	★★★★★	
Content Type Counter	<input type="checkbox"/>	★★★★★	

Yara

This extension allows you to perform on-demand Yara scans of websites within the Burp interface based on custom Yara rules that you write or obtain. Example use cases include scanning spidered sites for obfuscated Javascript or any other specific string patterns of interest present in any part of a request or response. It has been tested with Yara 3.4 on Windows 7 and 10, and Kali 2.0.

Requires the latest version of the standalone Yara binary (3.4) for your OS. Instructions at: <https://github.com/plusvic/yara/releases/tag/V3.4.0>.

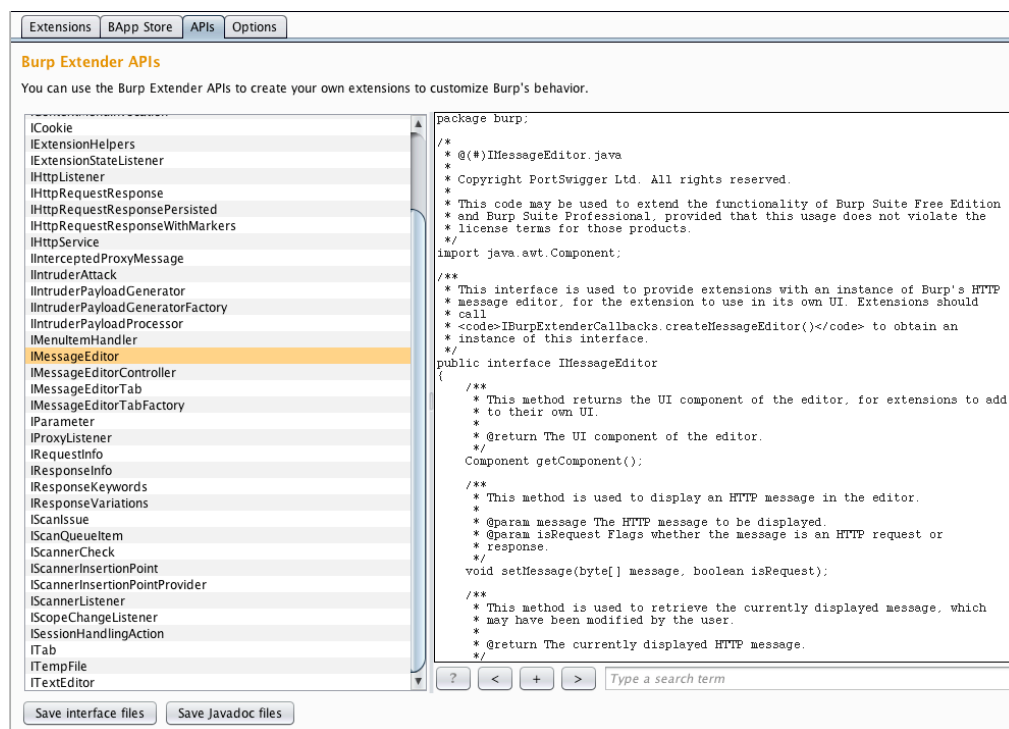
Requires Jython version 2.7 or later.

Author: Ian Duffy
Version: 1.0
Rating: ★★★★★

To use Python extensions, you need to download Jython, and configure its location in Burp Extender options.

Extender GUI 3/4

- Java APIs reference
- <https://portswigger.net/burp/extender/api/>



Extensions BApp Store APIs Options

Burp Extender APIs

You can use the Burp Extender APIs to create your own extensions to customize Burp's behavior.

- ICookie
- IExtensionHelpers
- IExtensionStateListener
- IHttpListener
- IHttpRequestResponse
- IHttpRequestResponsePersisted
- IHttpRequestResponseWithMarkers
- IHttpService
- IInterceptedProxyMessage
- IIntruderAttack
- IIntruderPayloadGenerator
- IIntruderPayloadGeneratorFactory
- IIntruderPayloadProcessor
- IMenuItemHandler
- IMessageEditor**
- IMessageEditorController
- IMessageEditorTab
- IMessageEditorTabFactory
- IParameter
- IProxyListener
- IRequestInfo
- IResponseInfo
- IResponseKeywords
- IResponseVariations
- IScanIssue
- IScanQueueItem
- IScannerCheck
- IScannerInsertionPoint
- IScannerInsertionPointProvider
- IScannerListener
- IScopeChangeListener
- ISessionHandlingAction
- ITab
- ITempFile
- ITextEditor

```
package burp;  
  
/**  
 * @(#)IMessageEditor.java  
 *  
 * Copyright PortSwigger Ltd. All rights reserved.  
 *  
 * This code may be used to extend the functionality of Burp Suite Free Edition  
 * and Burp Suite Professional, provided that this usage does not violate the  
 * license terms for those products.  
 */  
import java.awt.Component;  
  
/**  
 * This interface is used to provide extensions with an instance of Burp's HTTP  
 * message editor, for the extension to use in its own UI. Extensions should  
 * call  
 * <code>IBurpExtenderCallbacks.createMessageEditor()</code> to obtain an  
 * instance of this interface.  
 */  
public interface IMessageEditor  
{  
    /**  
     * This method returns the UI component of the editor, for extensions to add  
     * to their own UI.  
     *  
     * @return The UI component of the editor.  
     */  
    Component getComponent();  
  
    /**  
     * This method is used to display an HTTP message in the editor.  
     *  
     * @param message The HTTP message to be displayed.  
     * @param isRequest Flags whether the message is an HTTP request or  
     * response.  
     */  
    void setMessage(byte[] message, boolean isRequest);  
  
    /**  
     * This method is used to retrieve the currently displayed message, which  
     * may have been modified by the user.  
     *  
     * @return The currently displayed HTTP message.  
     */  
}
```

? < + > Type a search term

Save interface files Save Javadoc files

Extender GUI 4/4

- Runtime settings
 - Automatic reload
 - Execution environments, dependencies, etc.

The screenshot displays the 'Options' tab in the Extender GUI. It features a navigation bar with 'Extensions', 'BApp Store', 'APIs', and 'Options'. The main content area is divided into sections for different programming environments:

- Settings:** A checkbox labeled 'Automatically reload extensions on startup' is checked.
- Java Environment:** A text input field for 'Folder for loading library JAR files (optional)' is followed by a 'Select folder ...' button.
- Python Environment:** A text input field for 'Location of Jython standalone JAR file:' is followed by a 'Select file ...' button. Below it, another text input field for 'Folder for loading modules (optional):' is followed by a 'Select folder ...' button.
- Ruby Environment:** A text input field for 'Location of JRuby JAR file:' is followed by a 'Select file ...' button.

In a nutshell, we can write code to:

- Analyze, tamper and reply HTTP requests/responses for all Burp tools
- Customize Intruder and Scanner payloads
- Modify Burp Suite's configuration
- Initiate actions like scanning and spidering
- Access runtime data
- Create custom UI tabs and context menu items

Java Interfaces Summarized 1/2

Core

- IBurpExtender
- IBurpExtenderCallbacks
- IExtensionStateListener
- IHttpListener
- IScopeChangeListener
- ISessionHandlingAction

Helpers

- IExtensionHelpers
- ITempFile

System-wide UI

- IContextMenuFactory
- IContextMenuInvocation
- IMenuItemHandler
- IMessageEditor
- IMessageEditorController
- IMessageEditorTab
- IMessageEditorTabFactory
- ITab
- ITextEditor

HTTP objects

- ICookie
- IHttpRequestResponse
- IHttpRequestResponsePersisted
- IHttpRequestResponseWithMarkers
- IHttpService
- IParameter
- IRequestInfo
- IResponseInfo
- IResponseKeywords
- IResponseVariations

Java Interfaces Summarized 2/2

Collaborator

- IBurpCollaboratorClientContext
- IBurpCollaboratorInteraction

Intruder

- IIintruderAttack
- IIintruderPayloadGenerator
- IIintruderPayloadGeneratorFactory
- IIintruderPayloadProcessor

Scanner

- IScanIssue
- IScannerCheck
- IScannerInsertionPoint
- IScannerInsertionPointProvider
- IScannerListener
- IScanQueueItem

Proxy

- IInterceptedProxyMessage
- IProxyListener

Javadoc for the win

- <https://portswigger.net/burp/extender/api/>

Package burp

Interface Summary
Interface
IBurpCollaboratorClientContext
IBurpCollaboratorInteraction
IBurpExtender
IBurpExtenderCallbacks
IContextMenuFactory
IContextMenuInvocation
ICookie
IExtensionHelpers
IExtensionStateListener
IHttpListener
IHttpRequestResponse
IHttpRequestResponsePersisted
IHttpRequestResponseWithMarkers
IHttpService
IInterceptedProxyMessage
IIntruderAttack
IIntruderPayloadGenerator
IIntruderPayloadGeneratorFactory
IIntruderPayloadProcessor
IMenuItemHandler

public interface *IBurpExtender*

- All extensions must implement this interface
- Implementations must be called **BurpExtender**, in the package **burp**, must be declared **public**, and must provide a **default (public, no-argument) constructor**
- The following method is invoked when the extension is loaded and provides callbacks

```
void registerExtenderCallbacks (IBurpExtenderCallbacks callbacks)
```



The background features a dark grey field with scattered white binary code (0s and 1s) at the top. On the left side, there is a large orange triangle pointing towards the center. On the right side, there is a large, stylized grey arrow pointing towards the center.

Extension Templates

Netbeans, Eclipse, IDEA

Project Templates 1/2

- To facilitate development and debugging, you can use our empty extension template:
 - <https://github.com/doyensec/burpdeveltraining/tree/master/BurpExtensionTemplate>
 - Available for Netbeans, Eclipse and IDEA

Project Templates 2/2

- The template includes:
 - *burp.BurpExtender* class
 - Setup for executing Burp's Main (*burp.StartBurp*) together with our extension - which makes debugging make better!
 - No need to import all *Interfaces*
 - Since we are referencing the original Burp's JAR

Netbeans

1. Open NetBeans
2. *File* → *Open Project* and select the / *BurpExtensionTemplate/Netbeans* folder from git repo. Click *Open*
3. Click *Resolve Problems*. Then *Resolve* and select your local copy of the Burp's JAR. Click *Close*.
4. You can now click *Run*

Eclipse 1/2

1. Open Eclipse
2. *File* → *Open Project* (or *Import...*)
3. Choose “Select an import source:” *Existing Projects into Workspace*. Click *Next*
4. Click *Browse*, and select the / *BurpExtensionTemplate/Eclipse* folder from git repo. Click *Ok* and then *Finish*
5. On the project Name (top left), right click and select *Properties*. Go to *Java Build*

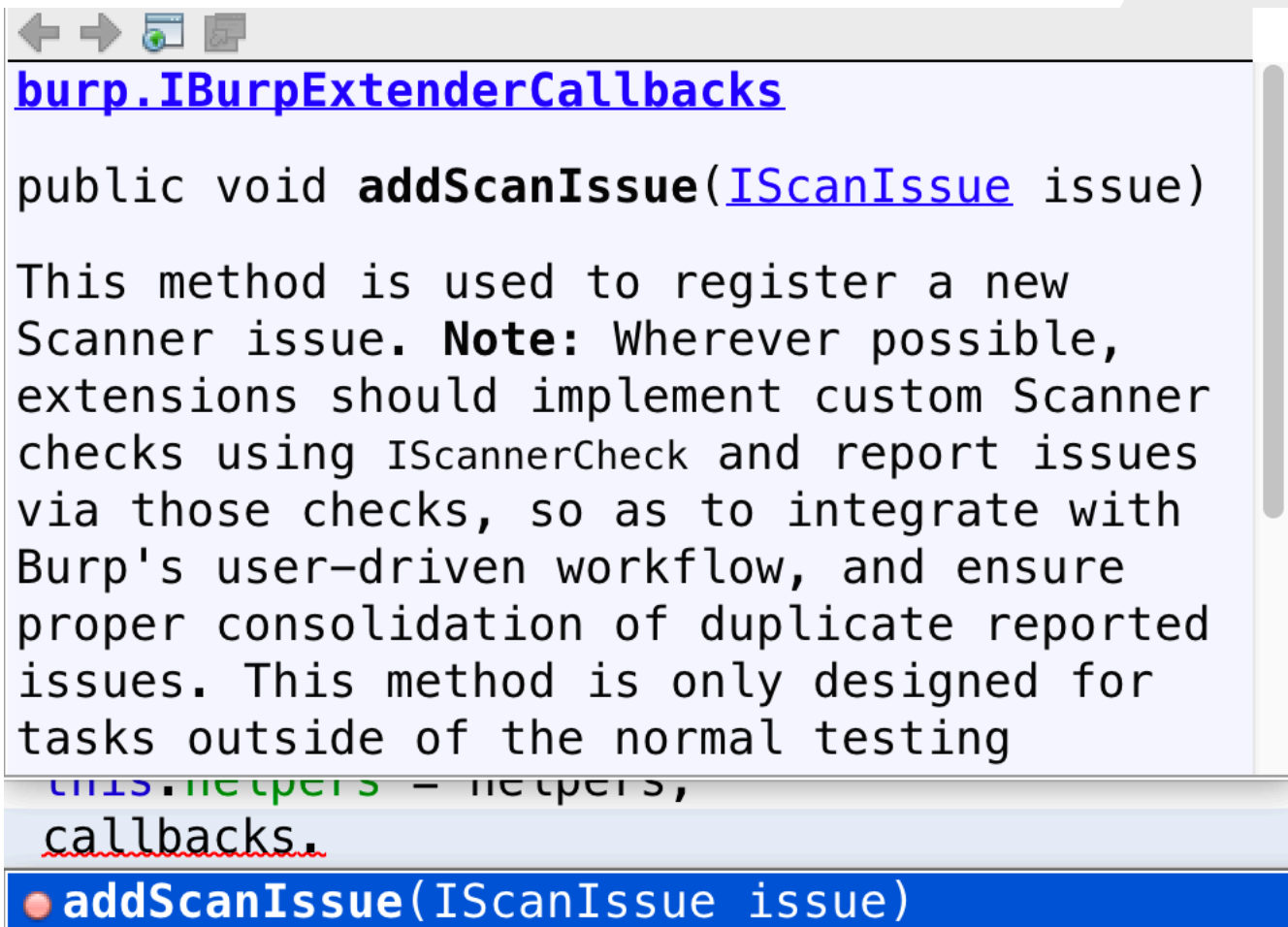
Eclipse 2/2

6. In the *Libraries* tab, click on the Burp JAR reference and click on *Edit*. Then, select your local copy of the Burp's JAR and click *Ok*
7. **If you're NOT using JavaSE-1.8**, you may also need to fix the runtime environment. In the *Libraries* tab, remove the *JRE System Library [JavaSE-1.8]*. Then, click on *Add Library* → *JRE System Library* → *Workspace Default JRE*. Click *Ok* and *Finish*
8. You can now click *Run*

IDEA

1. Open IntelliJ IDEA
2. Select *Open* and choose the / *BurpExtensionTemplate/IDEA* from git repo. Click *Open*
3. *File* → *Project Structure*. Select the Burp's JAR reference (in red). Right-Click and choose *Edit*..
4. Click *+* and select your local copy of the Burp's JAR. Click *Ok*. Select the old reference (in red) and Click *-* to remove the item
5. You can now click *Run*

Pro Tip: Burp JavaDocs



The screenshot shows a web browser window with the following content:

- Navigation icons: back, forward, home, and refresh.
- URL: [burp.IBurpExtenderCallbacks](#)
- Method signature: `public void addScanIssue(IScanIssue issue)`
- Description: This method is used to register a new Scanner issue. **Note:** Wherever possible, extensions should implement custom Scanner checks using `IScannerCheck` and report issues via those checks, so as to integrate with Burp's user-driven workflow, and ensure proper consolidation of duplicate reported issues. This method is only designed for tasks outside of the normal testing
- Additional text: `this.helpers = helpers,`
[callbacks.](#)
- Method signature in a blue bar: `addScanIssue(IScanIssue issue)`

Add Burp JavaDocs to Netbeans

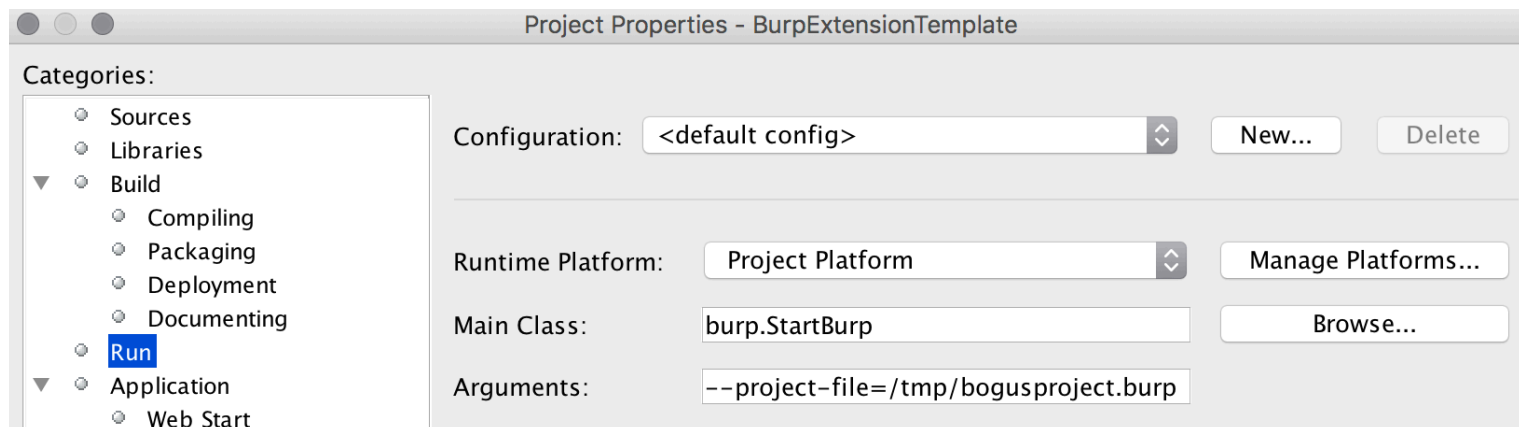
```
private final burp.  
private final IBurpExtenderCallbacks  
  
public SiteLoggerPanel( IBurpExtenderCallbacks callbacks, IExtensionHelpers helpers )  
{  
    initComponents();  
    this.callbacks = callbacks;  
    this.helpers = helpers;  
    callbacks.  
}  
  
/**  
 * This method is called  
 * WARNING: Do NOT modify  
 * regenerated by the For  
 */  
@SuppressWarnings("unchecked")  
Generated Code
```

1. Click on "Attach Javadoc..."
2. Select Burp JAR

```
public SiteLoggerPanel( IBurpExtenderCallbacks callbacks, IExtensionHelpers helpers )  
{  
    initComponents();  
    this.callbacks = callbacks;  
    this.helpers = helpers;  
    callbacks.  
}  
  
/**  
 * This method is called  
 * WARNING: Do NOT modify  
 * regenerated by the For  
 */  
@SuppressWarnings("unchecked")  
Generated Code
```

Pro Tip: Avoiding Project Options At Startup

- If you're debugging your extension, Burp startup project options will slow you down
- You can disable it, by adding the `--project-file` argument. Burp will generate a new project file with default configs





Our First Extension

Hello Burp!

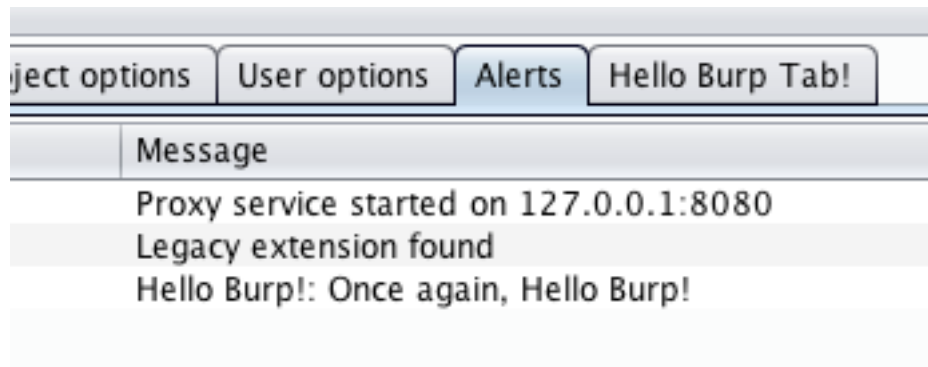
Let's write some code...

- In *BurpExtender.java*, let's customize the *registerExtenderCallbacks* method to implement our **Hello Burp!** extension
- We want to:
 - Issue an alert
 - Write to StdIn and StdOut
 - Create a new UI component (Tab)

Alerts

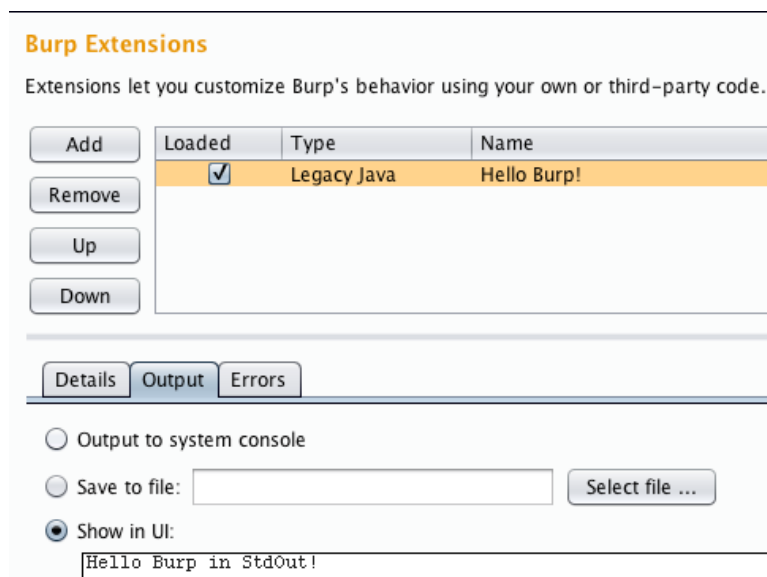
- Useful for info, error and other user notifications

```
callbacks.issueAlert("Hello Burp!");
```



Stdout and Stderr

- Stdout and Stderr can be customize from Burp's Extension UI, and it's transparent
- Simple use a *PrintWriter* on *callbacks.getStdout()*



Burp Extensions

Extensions let you customize Burp's behavior using your own or third-party code.

	Loaded	Type	Name
<input checked="" type="checkbox"/>	Legacy Java	Hello Burp!	

Output configuration:

Output to system console

Save to file:

Show in UI:

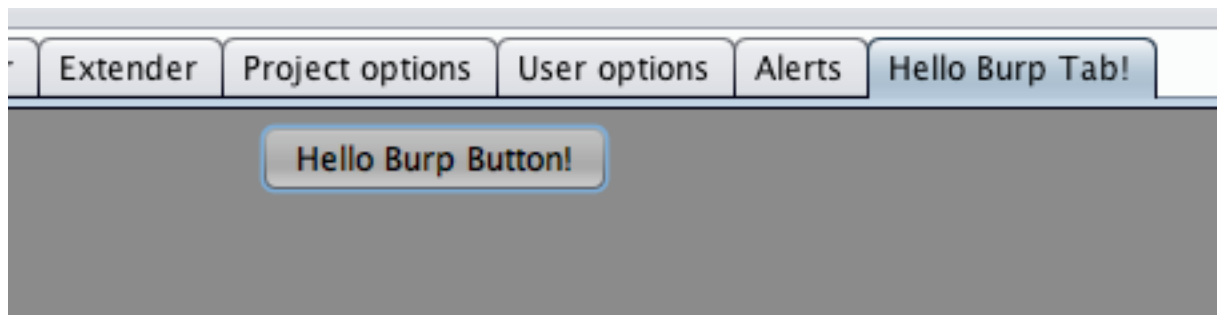
```
Hello Burp in StdOut!
```


Custom Tab and UI Components

- Create a class that *implements ITab* and implement the *getTabCaption()* and *getUiComponent()*
- *Pro Tip:* Use *callbacks.customizeUiComponent()* to adjust your custom UI with Burp's style, including font size, colors, table line spacing, etc.

Our custom Hello Burp AWT Component

- AWT coding is fun!




Coding Time!


- We will do this together!




Code Complete Extension


- <https://github.com/doyensec/burpdeveltraining/tree/master/HelloBurp>
- To run this exercise in Python/Ruby, remember to set the *Environment* option

 **Python Environment**

 These settings let you configure the environment for executing extensions that are written

Location of Jython standalone JAR file:

 **Ruby Environment**

 These settings let you configure the environment for executing extensions that are written

Location of JRuby JAR file:

Exercise

- Consult <https://portswigger.net/burp/extender/api/> and extend our HelloBurp extension:
 - 1.To issue an alert displaying Burp version number
 - 2.Unload the extension



Building a custom logger

SiteLogger

Requirements

- Save all HTTP Requests and Responses for a specific site
- Save all scan results (active and passive) for the same site
- Persistent storage using MongoDB
- Easy to use tool. It should be possible to setup and save from a UI

Plugin Structure

burp package

BurpExtender

registerExtenderCallbacks()

Your package

SiteLoggerTab implements ITab

getTabCaption()
getUiComponent()

SiteLoggerPanel

Note:

For Python/Ruby, a single file with three public classes is used.

getSiteMap and getScanIssues

- `IHttpRequestResponse []`
`getSiteMap (java.lang.String urlPrefix)`
- `IScanIssue []`
`getScanIssues (java.lang.String urlPrefix)`
- `urlPrefix` - This parameter can be used to specify a URL prefix, in order to extract a specific subset of the site map/findings

MongoDB

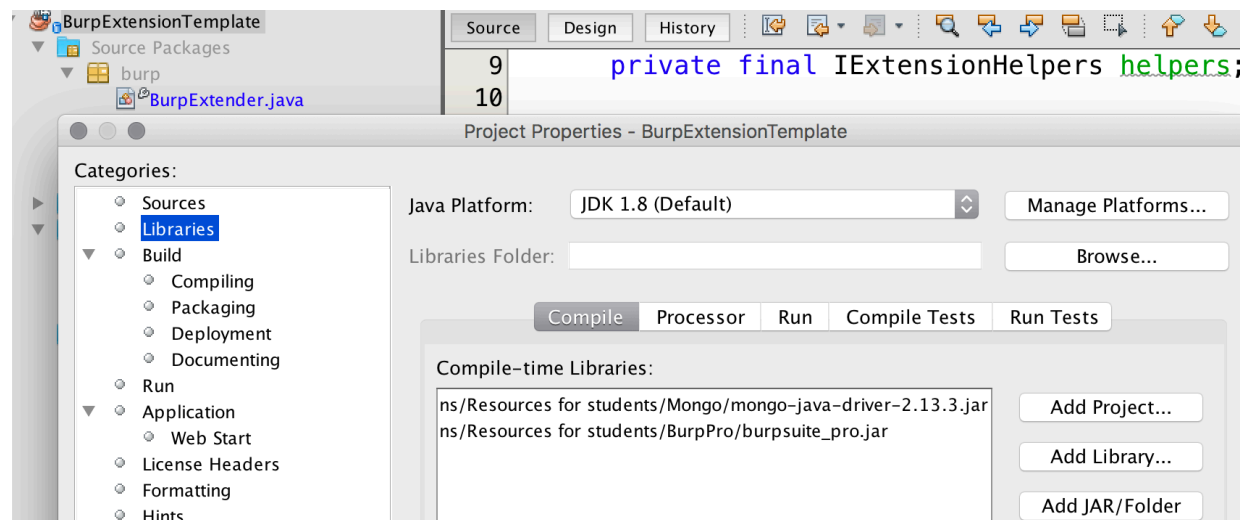
- There's a MongoDB instance running on **27017/tcp** (no authentication required)
- It's a shared environment - be nice!
- Please name your db **sitelogger_<yourName>**
- For a quick intro to MongoDB with Java:
 - <http://www.mkyong.com/mongodb/java-mongodb-hello-world-example/>

MongoDB Driver

- **[Java]** Download *mongo-java-driver-2.13.3.jar*
- **[Ruby]** Set *Extender->Options->JavaEnvironment* “Folder for loading library JAR files” to a folder containing *mongo-java-driver-2.13.3.jar*
- **[Python]** Set *Extender->Options->PythonEnvironment* “Folder for loading modules” to a folder containing *pymongo*

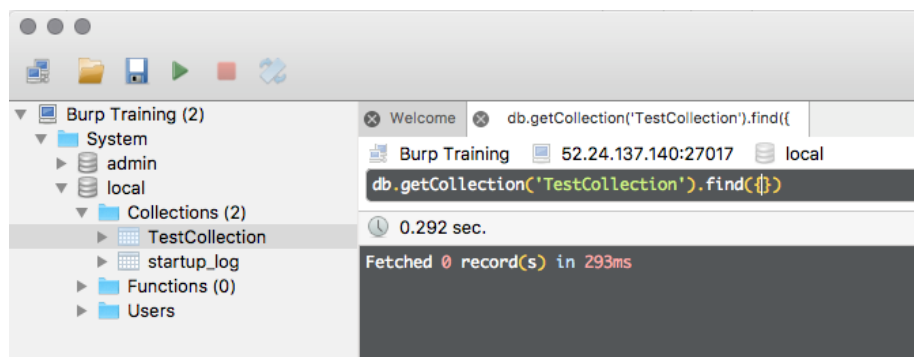
Add MongoDB Driver to the IDE

1. Select Project Name
2. Right-click on 'Properties'
3. Select 'Libraries'
4. Add JAR/Folder



MongoDB Client

- You can use Robomongo



SiteLogger UI

Burp Intruder Repeater Window Help

Repeater	Sequencer	Decoder	Comparer	Extender
Target	Proxy	Spider	Scanner	Intruder
Project options	User options	Alerts	SiteLogger	

Website:

MongoDB Host:

MongoDB Port:

Coding Time!



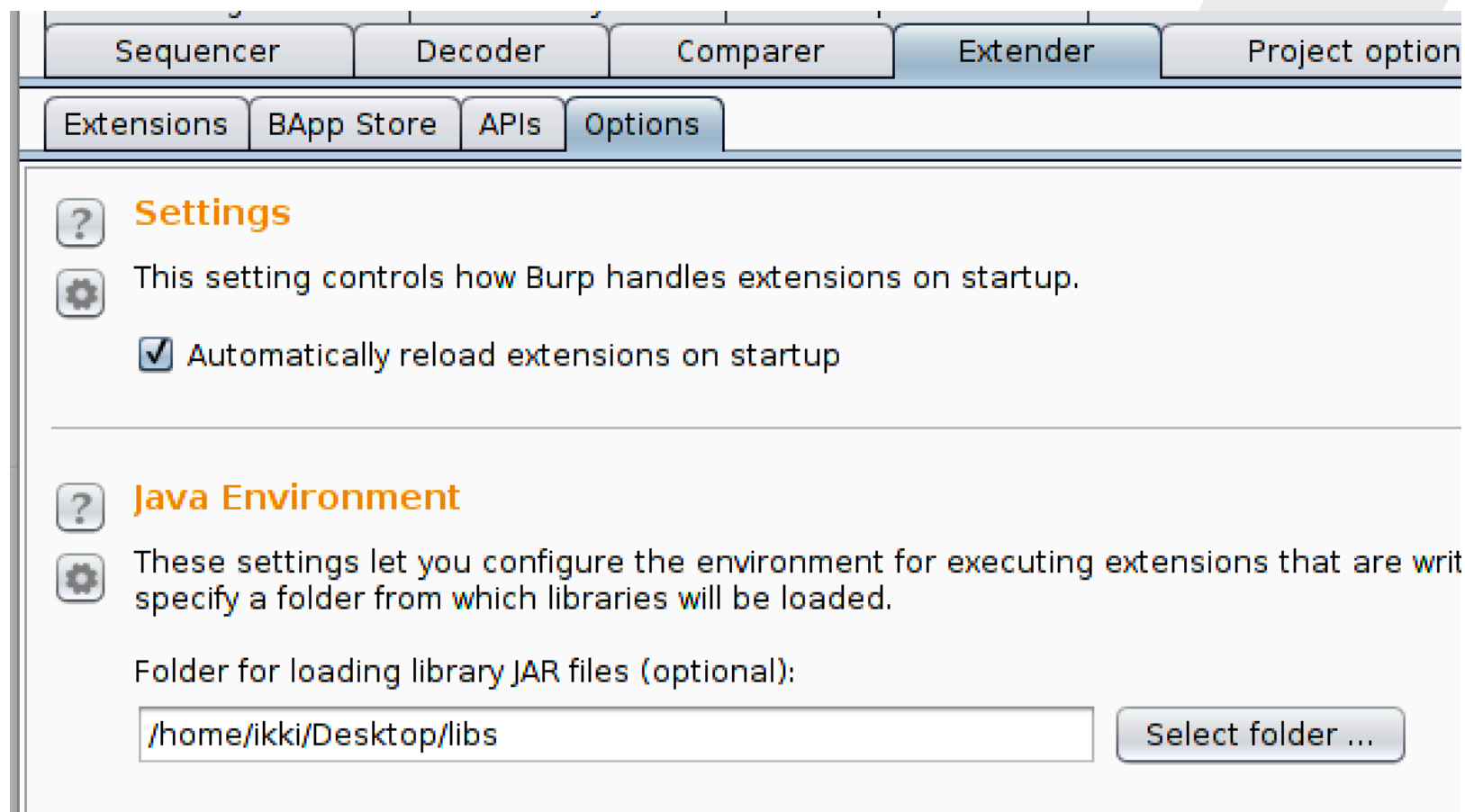
1. Let's review together the basic skeleton using WIP1 files
2. Let's build together the *swing.JPanel* using Netbeans WYSIWYG editor
3. Implement the remaining logic (~45 mins)
4. We will complete the exercise together

Extension dependencies

```
java.lang.NoClassDefFoundError: com/mongodb/MongoClient
    at
com.doyensec.sitelogger.SiteLoggerPanel.logButtonActionPerformed(SiteLoggerPanel.java:121)
    at
com.doyensec.sitelogger.SiteLoggerPanel.access$000(SiteLoggerPanel.java:21)
...
```

- How can we import extension libs?
 - (a) Setup Burp Java Environment
 - (b) Embed libs within the extension JAR

(a) Burp Java Environment



The screenshot shows the Burp Suite interface with the 'Extender' tab selected. Below the tabs, the 'Options' sub-tab is active, displaying the 'Settings' and 'Java Environment' sections.

Sequencer **Decoder** **Comparer** **Extender** **Project options**

Extensions **BApp Store** **APIs** **Options**

Settings

This setting controls how Burp handles extensions on startup.

Automatically reload extensions on startup

Java Environment

These settings let you configure the environment for executing extensions that are written in Java. You can specify a folder from which libraries will be loaded.

Folder for loading library JAR files (optional):

(b) Embed libs within the extension JAR

- Build.xml for the rescue!

```
<target name="-post-jar">
  <copy file="resources/README.txt" flatten="true" todir="${dist.dir}"/>
  <jar jarfile="dist/${ant.project.name}_v0.3.1.jar">
    <zipfileset src="${dist.jar}" excludes="META-INF/*" />
    <zipfileset src="lib/mongo-java-driver-2.13.3.jar" excludes="META-INF/*" />
    <fileset dir="dist">
      <include name="README.txt"/>
    </fileset>
  </jar>
  <jar basedir="src" destfile="dist/${ant.project.name}_v0.3.1-sources.jar"/>
</target>
```

Code Complete Extension

- <https://github.com/doyensec/burpdeveltraining/tree/master/SiteLogger>

Building a replay tool

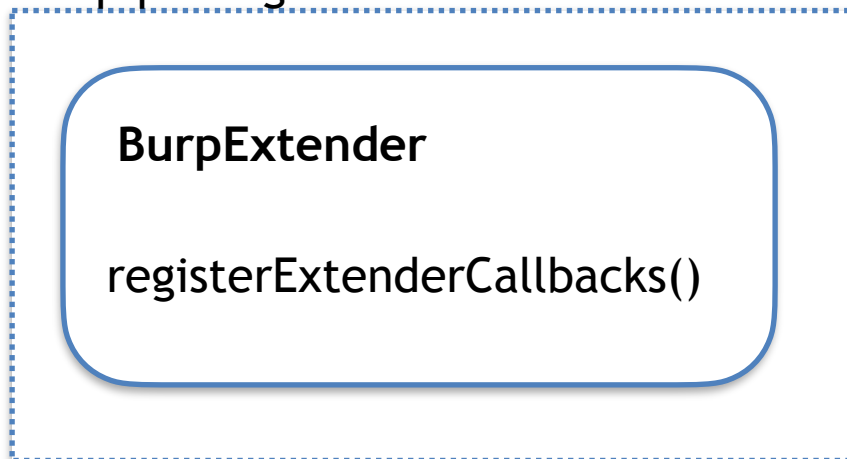
ReplayAndDiff

Requirements

- From the database, retrieve a login HTTP request with credentials in order to obtain a fresh session
- Issue the login request and add the new cookie to Burp's Cookies Jar
- Replay a scan on the site previously saved by SiteLogger
- Compare results and generate scan report, if the new scan includes new findings

Plugin Structure

burp package



- This extension should be executed in headless mode
- At startup, add the following JVM flag:
`-Djava.awt.headless=true`

Parse Command Line Args

- `java.lang.String[]`
`getCommandLineArguments()`
- `String[]` can be parsed with an external library (e.g. JSAP, Apache Commons CLI, etc.). For now, let's use a simple loop:

```
String[] args = callbacks.getCommandLineArguments();
for (String arg: args) {
    if(arg.contains("-h=") || arg.contains("--host=")) {
        MONGO_HOST = arg.substring(arg.indexOf('=')+1);
    }else if {
...

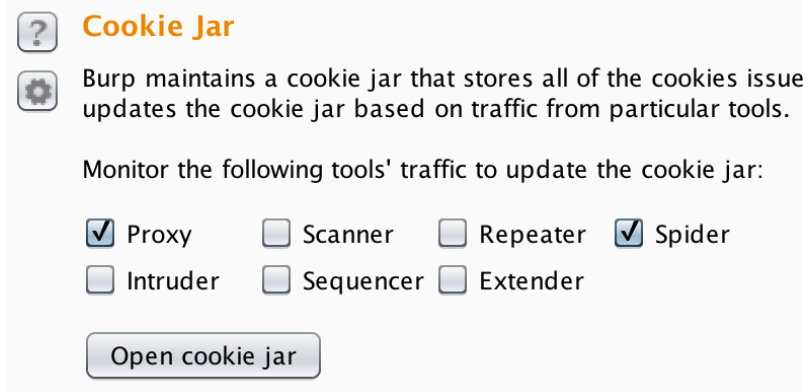
```

makeHttpRequest and analyzeResponse

- `byte []`
`makeHttpRequest (java.lang.String host, int port, boolean useHttps, byte [] request)`
- `IResponseInfo`
`analyzeResponse (byte [] response)`
- `IResponseInfo` **contains key details about an HTTP response including cookies, using `getCookies ()`**

Burp's Cookie Jar

- Burp maintains a cookie jar that stores all of the cookies issued by sites you visit
- The cookie jar is shared between all of Burp's tools
- `void
updateCookieJar(ICookie
cookie)`



Run a scan

- Two types of scan:
 - `IScanQueueItem doActiveScan(java.lang.String host, int port, boolean useHttps, byte[] request)`
 - `void doPassiveScan(java.lang.String host, int port, boolean useHttps, byte[] request, byte[] response)`
 - **Passive checks are executed immediately on stored request/response pairs, while Active checks require live traffic**
 - `IScanQueueItem` **can be used to check the status of the queue items**

Caveat

- When executing an active scan, Burp expects all target URLs to be in scope
- If the request is NOT within the current active scanning scope, the user will be asked if they wish to proceed with the scan
- Since we're running in headless mode, make sure to add:
- ```
void includeInScope (java.net.URL url)
```

# Diffing

- For this plugin, let's adopt a simple heuristic

```
for (IScanIssue finding : allVulns) {
 //Search in MongoDB for a finding with the
 same type, name and URL
 //If cursor.size() == 0, we have a new
 finding!
```

- If there is at least one new finding in the new scan,  
generate the report

```
void generateScanReport(java.lang.String format,
 IScanIssue[] issues,
 java.io.File file)
```

# Login Request Setup 1/4

1. With Burp enabled, perform a login to the page
2. Go the request, and right-click to open the contextual menu. Select "Save Item"
3. Open the saved file, and copy the Base64 request

# Login Request Setup 2/4

```
POST /burp.php HTTP/1.1
Host: 52.24.137.140
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.
Accept: text/html,application/xhtml+xml,application/xml
Accept-Language: en-US,en;q=0.5
Referer: http://52.24.137.140/
Content-Type: application/x-www-form-urlencoded
Content-Length: 24
Cookie: sessionId=fac1323fee9252af2f
Connection: close
Upgrade-Insecure-Requests: 1

email=test&password=test
```

- Send to Spider
- Do an active scan
- Do a passive scan
- Send to Intruder
- Send to Repeater
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Show response in browser
- Request in browser
- Engagement tools
- Copy URL
- Copy as curl command
- Copy to file
- Save item

# Login Request Setup 3/4

- `<request base64="true"><!`

```
[CDATA[UE9TVCAvYnVycC5waHAgSFRUUC8xLjENCkhvc3Q6IDU
yLjI0LjEzNy4xNDANCIVzZXItQWdlbnQ6IE1vemlsbGEvNS4wIChN
YWNpbmRvc2g7IEludGVsIE1hYyBPUyBYIDewLjEyOyBydjo1My4
wKSBHZWNrby8yMDEwMDEwMDEwMDEwMDEwMDEwMDEwMDEwMDEwMDEw
VwdDogdGV4dC9odG1sLGFwcGxpY2F0aW9uL3hodG1sK3htbC
xhcHBsaWNhdGlvbi94bWw7cT0wLjksKi8qO3E9MC44DQpBY2N
lcHQqTGFuZ3VhZ2U6IGVuLVVTLGVuO3E9MC41DQpSZWZlcmV
yOiBodHRwOi8vNTluMjQuMTM3LjE0MC8NCkNvbnRlbnQtVHlw
ZTogYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkd
QpDb250ZW50LUxlbmd0aDogMjUwMDEwMDEwMDEwMDEwMDEwMDEw
kPWZhdGEzZm9udGVuZm9udGVuZm9udGVuZm9udGVuZm9udGVuZm
c2UNCIVwZ3JhZGUtSW5zZWN1cmUtUmVxdWVzdHM6IDENCg
0KZW1haWw9dGVzdCZwYXNzd29yZD10ZXN0]]></request>
```

# Login Request Setup 4/4

- Using Robomongo, create a new collection named “login”
- Then, insert the following entry:

```
use sitellogger
db.login.insert({
 host: '52.24.137.140',
 port: NumberInt(80),
 protocol: 'http',
 request:
'UE9TVCAvYnVycC5waHAgSFRUUC8xLjENCkhvc3Q6IDUyLjI0LjEzNy4xNDANC1VzZXItQWdlbnQ6I
E1vemlsbGEvNS4wIChNYWNpbmRvc2g7IEludGVsIE1hYyBPUyBYIDewLjEyOyBydjo1My4wKSBHZWN
rby8yMDEwMDEwMSBGaXJlZm94LzUzLjANCkFjY2VwdDogdGV4dC9odG1sLGFwcGxpY2F0aW9uL3hod
G1sK3htbCxxhcHBsaWNhdGlvbi94bWw7cT0wLjksKi8q03E9MC44DQpBY2NlcHQtTGFuZ3VhZ2U6IGV
uLVVTLGVu03E9MC41DQpSZWZlcmVyOiBodHRw0i8vNTIuMjUyYyZg0KQ29ubmVjdGlvbjogY2xvc2UNC1VwZ
TogYXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVkdQpDb250ZW50LUxlbmd0aDogMjQNCkN
vb2tpZTogc2Vzc2lvcmlkPWZhYzEzMjUyYyZg0KQ29ubmVjdGlvbjogY2xvc2UNC1VwZ
3JhZGUtSW5zZWN1cmUtUmVxdWVzdHM6IDENCg0KZW1haWw9dGVzdCZwYXNzd29yZD10ZXN0
'})
```



# Coding Time!



1. Let's review together the basic skeleton using WIP1 files
2. Implement the remaining logic (~1 hour)
3. We will complete the exercise together

# Code Complete Extension

- <https://github.com/doyensec/burpdeveltraining/tree/master/ReplayAndDiff>

# Building a custom Passive Scan check

DetectSRI

# Requirements

- Create an extension that detects whether the specific HTTP response does not use SubResource Integrity (SRI)
  - “Subresource Integrity (SRI) is a security feature that enables browsers to verify that files they fetch (for example, from a CDN) are delivered without unexpected manipulation. It works by allowing you to provide a cryptographic hash that a fetched file must match”
  - [https://developer.mozilla.org/en-US/docs/Web/Security/Subresource\\_Integrity](https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity)

# SubResource Integrity (SRI)

- Check the login page source

```
view-source:52.24.137.140/index.html
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5 <meta charset="utf-8" />
6 <title>Burp Fake Login</title>
7 <link href="assets/css/style.css" rel="stylesheet" integrity="sha384-
M1jJIGbFlKmkFRh+jO8S0YqR7EdK5pWfF07FqQNsutTtJV/204RBfRd9AbyfHKvUf" />
```

# Plugin Structure

burp package

```
BurpExtender
registerExtenderCallbacks()

callbacks.registerScannerCheck(this);
```

- In this simple passive check, we can include the heuristic code in the BurpExtender class itself

# doPassiveScan

- `java.util.List<IScanIssue>`  
`doPassiveScan(IHttpRequestResponse`  
`baseHttpRequestResponse)`
- Extensions should only analyze the HTTP messages provided during passive scanning

# IScanIssue interface

- We will return our custom implementation
- `getConfidence()`
  - "Certain", "Firm" or "Tentative"
- `getSeverity()`
  - "High", "Medium", "Low", "Information" or "False positive"
- `getIssueType()`
  - 0x08000000



# consolidateDuplicateIssues

- `int consolidateDuplicateIssues (IScanIssue existingIssue, IScanIssue newIssue)`
- **It's your responsibility to handle duplicates**
- **The Scanner will invoke this method if there are multiple issues for the same URL path**

# HTML Ascii Art anyone?

**i** Subresource Integrity (SRI) Missing

- i Frameable response (potential Clickjacking)
- i Path-relative style sheet import
- i HTML uses unrecognized charset

---

Advisory Request Response

---

**i** Subresource Integrity (SRI) Missing

---

Issue: Subresource Integrity (SRI) Missing  
Severity: Information  
Confidence: Certain  
Host: <https://doyensec.com>  
Path: /contact.html

---

**Note:** This issue was generated by the Burp extension: DetectSRI.

**Issue detail**

Burp Scanner has not identified Subresource Integrity (SRI) attributes in the following page: <https://doyensec.com:443/contact.html>

**Issue background**

Subresource Integrity (SRI) is a security feature that enables browsers to verify that files they fetch (for example, from a CDN) are delivered that a fetched file must match.

**Issue remediation**

This is an **informational** finding only.

# Coding Time!

- We will do this together!



# Code Complete Extension

- <https://github.com/doyensec/burpdeveltraining/tree/master/DetectSRI>

# Building a custom Active Scan check

DetectELJ

# Requirements

- Create an extension that detects **Expression Language** injection vulnerabilities
  - Type of injection that occurs when attackers control data that is evaluated by an Expression Language (EL) interpreter

# Expression Language (EL) injection

- Example:
  - $\${1336+1}$   $\rightarrow$  1337
- Think about Struts2 OGNL, Apache Jakarta, Spring's SPEL
- If interested, read <https://www.mindedsecurity.com/fileshare/ExpressionLanguageInjection.pdf>

# Plugin Structure

burp package

```
BurpExtender
```

```
registerExtenderCallbacks()
```

```
callbacks.registerScannerCheck(this);
```

- We can re-use the same passive scanner check skeleton, and instead implement `doActiveScan()`



# Understanding Response Changes

- Burp Extender APIs provides a useful helper method to analyze variations between two or more HTTP Responses
- `IResponseVariations`  
`analyzeResponseVariations(byte[] ... responses)`
- `IResponseVariations` is a list of `String` representing the specific attributes that changed between those responses
  - `content_length`
  - `whole_body_content`
  - and so many others

# IResponseVariations Attribute Types

- status\_code
- input\_image\_labels
- page\_title
- visible\_text
- button\_submit\_labels
- div\_ids
- word\_count
- content\_type
- outbound\_edge\_tag\_names
- whole\_body\_content
- etag\_header
- visible\_word\_count
- content\_length
- header\_tags
- tag\_ids
- comments
- line\_count
- set\_cookie\_names
- last\_modified\_header
- first\_header\_tag
- tag\_names
- input\_submit\_labels
- outbound\_edge\_count
- initial\_body\_content
- content\_location
- limited\_body\_content
- canonical\_link
- css\_classes
- location
- anchor\_labels

# Scan Issue

5 11:36:09 18 Mar 2017 Issue found  Expression Language (EL) Injection Detected https://ikkisoft.com /elj.php par parameter High Firm

Advisory

## Expression Language (EL) Injection Detected

Issue: Expression Language (EL) Injection Detected  
Severity: High  
Confidence: Firm  
Host: https://ikkisoft.com  
Path: /elj.php

**Note:** This issue was generated by the Burp extension: DetectELJ.

### Issue detail

Burp Scanner has identified an Expression Language injection in: <https://ikkisoft.com:443/elj.php?par=aaa>

### Issue background

Expression Language injections occur when input data is evaluated by an expression language interpreter. An attacker can read server-side data, such as the content of server-side variables, and some other inner configuration details.

### Issue remediation

Apply input validation best practices, and reject \$, #, { and other variations.

# Coding Time!



1. Let's review together the basic skeleton using WIP1 files
2. Implement the remaining logic (~30mins)
3. We will complete the exercise together

# Pro Tip 1/2

- When creating a new issue, we can also specify markers to show specific strings within requests/responses
- This is done using the following callback method:

```
IHttpRequestResponseWithMarkers
applyMarkers (IHttpRequestResponse
httpRequestResponse,
java.util.List<int []> requestMarkers,
java.util.List<int []> responseMarkers)
```

# Pro Tip 2/2

- Within our implementation of `IScanIssue`, we can customize the object returned by `getHttpMessages()`

```
@Override
public IHttpRequestResponse[] getHttpMessages() {
 String strRes = helpers.bytesToString(reqres.getResponse());
 int[] marks = new int[2];
 marks[0] = strRes.indexOf("1337");
 marks[1] = marks[0] + 4;
 List<int[]> marksList = new ArrayList<>(1);
 marksList.add(marks);
 IHttpRequestResponseWithMarkers reqresMark = callbacks.applyMarkers(reqres, null, marksList);
 IHttpRequestResponse[] rra = { reqresMark };
 return rra;
}
```

# Scan Issue Improved

```

<div id="protected-page">
 You are logged in</h1>

 <p>Dear Burp, I love you 1337 times!</p><a
2.168.100.111/burp.php?logout=true"
utton">Logout
</div>

```

Advisory
Request
Response

**!** **Expression Language (EL) Injection Detected**

---

Issue: **Expression Language (EL) Injection Detected**  
Severity: **High**  
Confidence: **Firm**  
Host: **http://192.168.100.111**  
Path: **/burp.php**

---

**Note:** This issue was generated by the Burp extension: DetectELJ.

**Issue detail**

Burp Scanner has identified an Expression Language injection in:  
**http://192.168.100.111:80/burp.php?elj=\$%7b1336%2b1%7d**

**Issue background**


Expression Language injections occur when input data is evaluated by an expression language interpreter. An attacker can read server-side data, such as the content of server-side variables, and some other inner configuration details.

**Issue remediation**

# Code Complete Extension

- <https://github.com/doyensec/burpdeveltraining/tree/master/DetectELJ>



The background is a dark grey color. At the top, there is a pattern of binary code (0s and 1s) in a light grey font. On the left side, there is a large orange triangle pointing right. On the right side, there are several overlapping, semi-transparent grey geometric shapes that resemble stylized arrows or chevrons pointing right.

# Building a security automation toolchain

## Jenkins + Burp

# Typical CI Environment

- Continuous Integration has become a widely adopted practice

The screenshot displays the Jenkins web interface for a pipeline named "My pipeline". The browser address bar shows "localhost:8082/view/Build%20pipeline/". The Jenkins header includes a search bar and the user "marcinp" with a "log out" link. The pipeline is currently at version 8 with no parameters. The pipeline steps are as follows:

- Test**: Jun 26, 2012 5:30:48 PM, 10 sec, marcinp
- Release**: Jun 26, 2012 5:31:03 PM, 12 sec
- Deploy to Test**: Jun 26, 2012 5:31:48 PM, 6 sec, marcinp
- Generate docs**: Jun 26, 2012 5:31:20 PM, 9.1 sec
- Deploy to Pre-Prod**: Pending
- Deploy to Prod**: Pending

Navigation icons for the pipeline include Run, History, Configure, Add Step, Delete, and Manage.


# Moving fast means testing fast

- We have a site logger
- We have an headless replay tool
- We have custom Scanner checks
- Let's integrate all components in a new *Build Step (SecTesting)* using **Jenkins** and the **Build Pipeline View** plugin

# Jenkins Setup 1/2

1. Download Jenkins - jenkins.war
2. Run it using `$java -jar jenkins.war`
3. Open the browser and visit <https://<HOST>:8080>
4. Type your auto-generated admin password
5. Select *Select Plugin to Install*
6. Search and install *Build Pipeline Plugin*

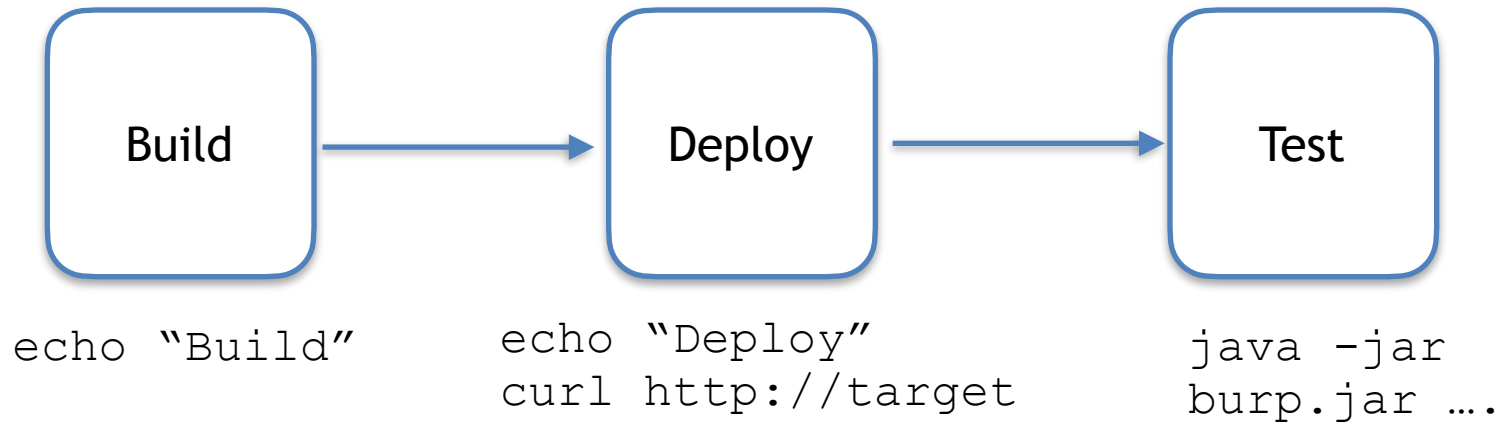
# Jenkins Setup 2/2

7. Coffee Break 
8. Create your admin user
9. It's time to create our build pipeline!

# Training Environment

- Multiple Jenkins instances already setup and ready to go
- Login with *admin:admin*

# Our pipeline plan



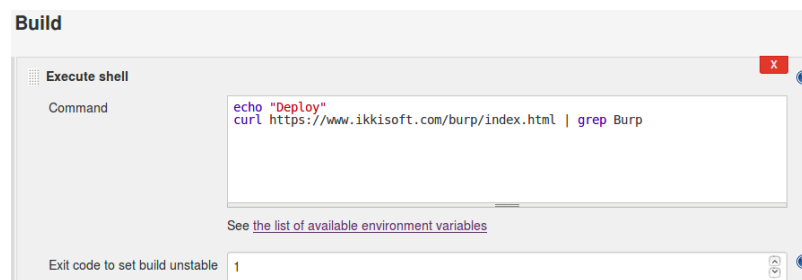
# (1) Build

- To simplify our setup, build is just simulated using a simple execute shell
- Click on “New Item”->”Freestyle Project” named “Build”
- In the Build section, select “Add Build Step” -> “Execute Shell”
- Type echo “Build”
- Save



## (2) Deploy

- To simplify our setup, deploy is also simulated
- Click on “New Item”->“Freestyle Project” named “Deploy”
- In the Build section, select “Add Build Step” -> “Execute Shell”
- Type `echo "Deploy"; curl http://127.0.0.1/index.html | grep Burp`
- Additionally, we can verify if the live endpoint is up and running with “Exit code to set build unstable”
  - If the exit code `!= 1` -> *Build Unstable*



- Finally, in “Build Triggers” select “Build after other projects are built” and type “Build”
- Save

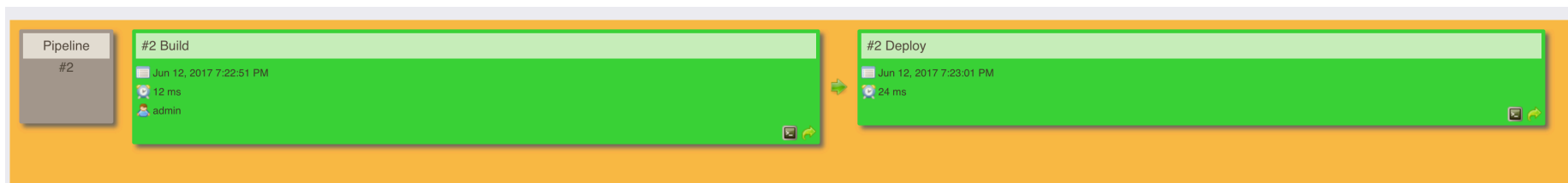
# Create New Build Pipeline View

- From the home, click on the +
- Select “Build Pipeline View” named “Security Pipeline” and then “Ok”
- (Optional) Type a description
- In the configuration “Select Initial Job”, pick “Build”
  - This is our first step for the overall execution

Upstream / downstream config

Select Initial Job

# Our Security Pipeline (so far)



- It's time to perform security testing on the “newly deployed code”

# (3) SecTesting

- A new task where we will execute Burp and our extensions

Build, deploy and test with Burp Scanner.

The screenshot displays a CI/CD pipeline interface with the following components:

- Runners:** Run, History, Configure, Add Step, Delete, Manage
- Pipeline #5:**
  - #5 Build:** Status: Success (Green). Duration: 0:18 sec. Executed on Mar 19, 2017 12:15:15 AM.
  - Deploy:** Status: N/A (Blue).
  - SecTesting:** Status: N/A (Blue).

# SecTesting Plan 1/2

1. Using Burp and our *SiteLogger* plugin, we have already recorded the traffic to our deployed application
2. With *ReplayAndDiff*, we can run a scan in headless mode, performing diff and generating a new findings report
3. Using DetectELJ and DetectedSRI, we can enhance our scanners capabilities

# SecTesting Plan 2/2

- Let's just put all together so that our build step will execute Burp with our plugins
- Additionally, we want to fail the build if there're new findings

# SecTesting Setup 1/4

- First, let's start Burp with the GUI and load all extensions. Then, close it gracefully.
  - In this way, Burp will reload all previously-loaded extensions at startup
  - We can use **\$ ssh -X** for X11 forwarding
    - *This step was already done by me*

# SecTesting Setup 2/4

- Create a new Freestyle project named “SecTesting”
- In the newly create SecTesting task, create a new “Execute Shell” with the following code:

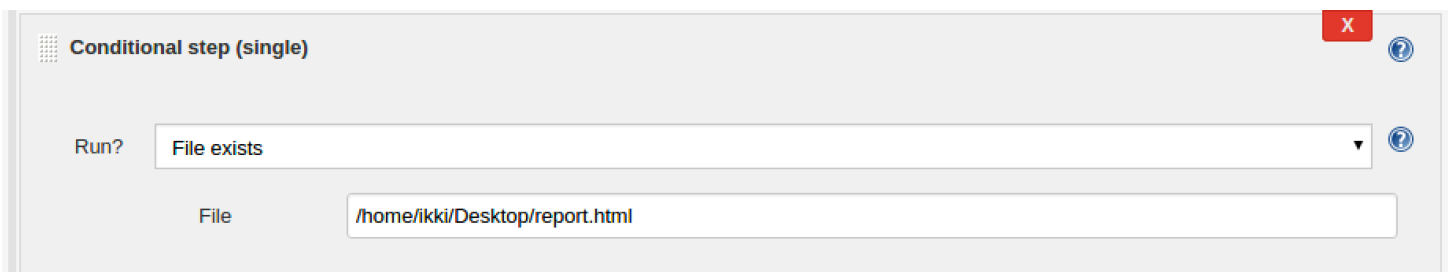
```
echo "Launching Burp Pro..."
```

```
java -Xmx256m -Djava.awt.headless=true
-jar /home/ubuntu/burpsuite_pro.jar
-h=127.0.0.1 -p=27017 -o=/home/ubuntu/
reports/ -r=report.html -t=40
```

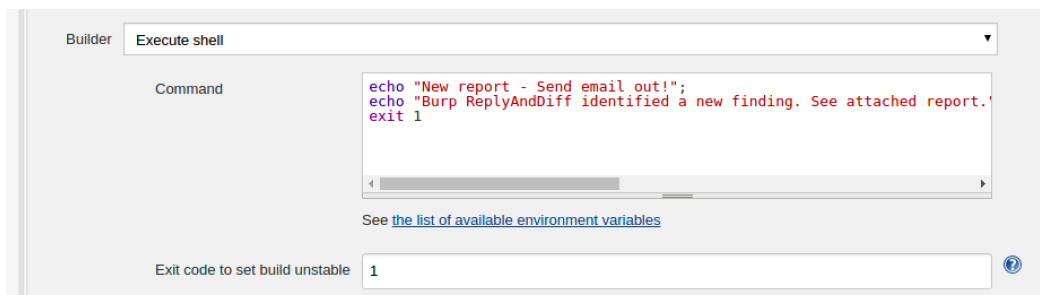


# SecTesting Setup 3/4

- Click “Add Build Step” and select “Conditional Step (single)”
- File: */home/ubuntu/reports/report.html*



- Then, “Execute Shell” again



# SecTesting Setup 3/4

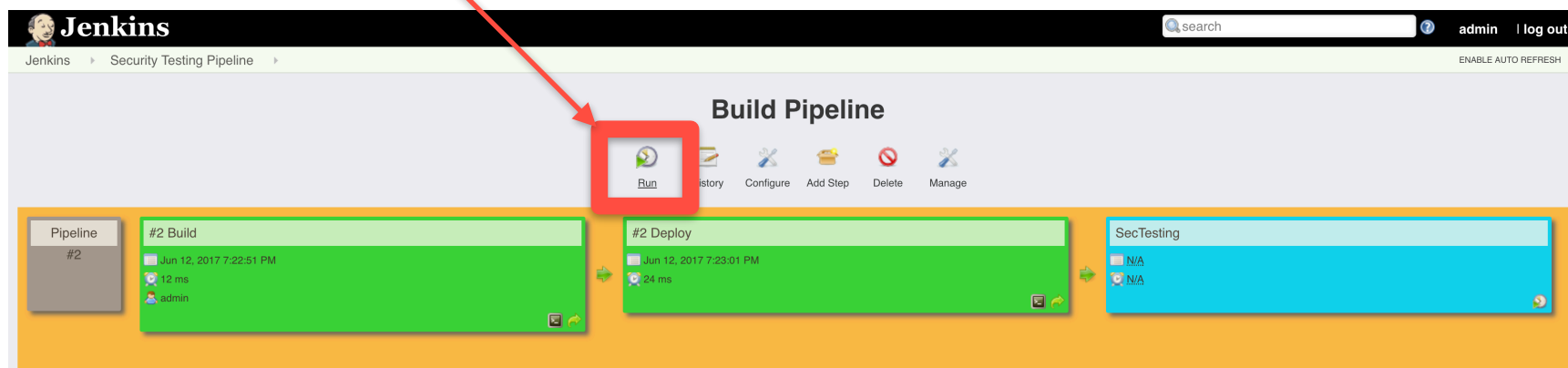
- As shell script use:

```
echo "New report - Send email out!";
echo "Burp ReplyAndDiff identified a new finding. See
attached report." | mail -A /home/ubuntu/reports/
report.html -s "Burp Scanner - New Finding" <email>
rm /home/ubuntu/reports/report.html
exit 1
```


- *Exit 1* will force the build as 'unstable'
  - In a real-world scenario, you could use Jenkins' Email Notification to send a custom email to the developers
- 
- In the configuration "Select Initial Job", pick "Build"
  - Save

# Ready To Go

- Click on Run to execute the entire pipeline



The screenshot shows the Jenkins web interface for a pipeline named 'Security Testing Pipeline'. The 'Build Pipeline' section is active, displaying a sequence of stages: '#2 Build' (green), '#2 Deploy' (green), and 'SecTesting' (blue). The '#2 Build' stage shows a timestamp of 'Jun 12, 2017 7:22:51 PM' and a duration of '12 ms'. The '#2 Deploy' stage shows a timestamp of 'Jun 12, 2017 7:23:01 PM' and a duration of '24 ms'. The 'SecTesting' stage shows 'N/A' for both timestamp and duration. A red box highlights the 'Run' button, which is a green play icon, and a red arrow points from the text above to this button. Other buttons like 'History', 'Configure', 'Add Step', 'Delete', and 'Manage' are also visible.

- Jenkins will display the progress. You can review the console output by clicking on the  symbol

# SecTesting Console Output

 Jenkins

Jenkins &gt; Security Testing Pipeline &gt; SecTesting &gt; #5

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output](#)**
- [View as plain text](#)
- [Edit Build Information](#)
- [Delete Build](#)
- [Previous Build](#)

## Console Output

```
Started by upstream project "Deploy" build number 5
originally caused by:
 Started by upstream project "Build" build number 5
originally caused by:
 Started by user admin
Building in workspace /home/ubuntu/jenkins/jenkins6/workspace/SecTesting
Run condition [File exists] enabling prebuild for step [Execute shell]
[SecTesting] $ /bin/sh -xe /tmp/hudson2616581029438597149.sh
+ echo Launching Burp Pro...
Launching Burp Pro...
+ java -Xmx256m -Djava.awt.headless=true -jar /home/ubuntu/burpsuite_pro.jar -h=127.0.0.1 -p=27017 -o=/home/ubuntu/reports/ -r=report.html -t=30
Proxy: Proxy service started on 127.0.0.1:8080

:: ReplayAndDiff Headless Extension ::

[*] Configuration {MONGO_HOST=127.0.0.1,MONGO_PORT=27017,OUTPUT_DIR=/home/ubuntu/reports/,REPORT_NAME=report.html,TIMEOUT=30}
Extender: DetectSRI: DetectSRI Passive Scanner check enabled
Extender: DetectELJ: DetectELJ Active Scanner check enabled
[*] Retrieving record for: 52.24.137.140
[*] Obtained cookie: sessionId:fac1323fee9252af2f
[!] Missing cookie attributes - e.g. domain not set
[*] Pausing extension...
[*] Resuming extension...
[*] Looking for: { "type": 5245344, "name": "Frameable response (potential Clickjacking)", "URL": "http://52.24.137.140:80/" }
[*] Got a new finding!
[*] Looking for: { "type": 3145984, "name": "Cleartext submission of password", "URL": "http://52.24.137.140:80/" }
[*] Got a new finding!
[*] Looking for: { "type": 5244928, "name": "Password field with autocomplete enabled", "URL": "http://52.24.137.140:80/" }
[*] Got a new finding!
[*] Looking for: { "type": 134217728, "name": "Subresource Integrity (SRI) Detected", "URL": "http://52.24.137.140:80/" }
[*] Got a new finding!
[*] Looking for: { "type": 16777728, "name": "Unencrypted communications", "URL": "http://52.24.137.140:80/" }
[*] Looking for: { "type": 2097960, "name": "Path-relative style sheet import", "URL": "http://52.24.137.140:80/" }
[*] Got a new finding!
```

# Moment of Truth

- Wait for the email :)

---

☆ ikki 

Burp Scanner - New Finding

To: luca@doyensec.com

---

Burp ReplyAndDiff identified a new finding. See attached report.



report.html

\*\* EXTRA MATERIAL \*\*

# Intruder Payloads Generator

Case Study: Bradamsa

# Intruder payloads

- Let's study the implementation of **Bradamsa**
- Radamsa for Burp
  - Original project:  
<https://github.com/ikkisoft/bradamsa>
  - Simplified code to study:  
<https://github.com/doyensec/burpdeveltraining/tree/master/Bradamsa>
- This extension provides a custom Intruder payload generator

# What is Radamsa

- Radamsa is a command-line test case generator for fuzzing
- It is scriptable and super easy to use

```
$ echo "aaa" | radamsa
:aaa
```



# Install Radamsa

```
$ git clone https://github.com/aoh/radamsa.git
```

```
$ cd radamsa
```

```
$ make
```

```
$ sudo make install
```

```
$ radamsa --help
```

- You need *gcc/clang*, *make* and *git*
- On Mac OS, change the Makefile to install in */usr/local/bin/*

# What is a payload generator?

## ? Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Va

Payload set:  Payload count: unknown

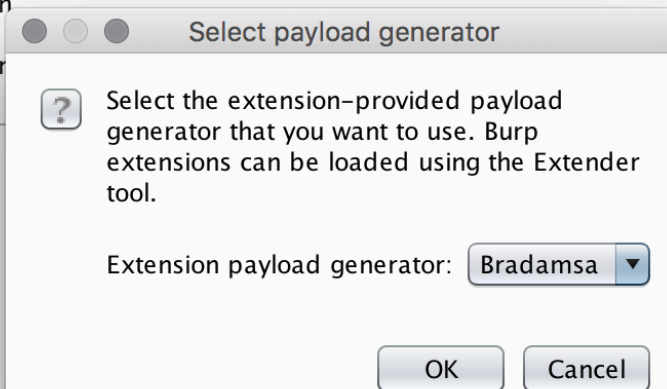
Payload type:  Request count: unknown

## ? Payload Options [Extension-generated]

This payload type invokes a Burp extension to generate payloads.

Selected generator: [NOT SELECTED]

Select generator ...



# Generator and Processor

- Extensions can register themselves as `registerIntruderPayloadGeneratorFactory` Or `registerIntruderPayloadProcessor`
- For payload generators, Burp expects the extension to return an implementation of the `IIntruderPayloadGenerator` interface

# In pseudo-code

```
class myGenerator implements IIntruderPayloadGenerator{

 @Override
 public boolean hasMorePayloads()
 { ... }

 @Override
 public byte[] getNextPayload(byte[] baseValue)
 { ... }

 @Override
 public void reset()
 { ... }
}
```

# Bradamsa Demo 1/2

Burp Intruder Repeater Window Help

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts Bradamsa

**Radamsa command line options**

For more details, please refer to the official Radamsa homepage (<https://code.google.com/p/ouspg/>)

Binary:

Count:

Output:

Seed:

Mutations:

Patterns:

Meta:

Delete sample files after execution Reset Settings

**Resulting Command Line:**

```
/usr/bin/radamsa -n 10 -o /var/folders/d6/zjpxvxxs21x6c4fgr9j1mrt00000gp/t/radamsa2446258691559248640/%n.out
```

# Bradamsa Demo 2/2

Patterns:

Meta:

Delete sample files after execution

**Resulting Command Line:**

Invalid mutations pattern [od,nd,bu]

# Let's have a look at the code

- <https://github.com/doyensec/burpdeveltraining/tree/master/Bradamsa>
- Focus on *RadamsaPayloadGenerator.java* and *BradamsaPanel.java*



# Conclusion



# Final Remarks

- Building extensions is **fun** and **useful** to improve efficacy and efficiency of security testing activities
- When analyzing custom protocols, Burp extensions make a big difference
- When Burp Scanner is integrated in the SDLC, creating custom check can ensure test coverage (regression, application-specific bugs, ...)

# What's next?

- Study the Burp API Javadoc
  - <https://portswigger.net/burp/extender/api/>
- Check the source code of BApp Store extensions on PortSwigger's Github
  - <https://github.com/PortSwigger>
- Build your extension!



Thank you!

Please email your feedback at  
[info@doyensec.com](mailto:info@doyensec.com)

Luca Carettoni - [luca@doyensec.com](mailto:luca@doyensec.com)